IDENTIFICATION

PRODUCT CODE: ZZ-ENSAA-7.0

PRODUCT TITLE: VAX DIAGNOSTIC SUPERVISOR

PRODUCT DATE: 18-JUN-1984

DEPARTMENT: BASE SYSTEM DIAGNOSTIC ENGINEERING

Diagnostic Supervisor release notes V5.3

A buffer in show device was increased to 256 characters.

Channel services was fixed to disable interrupts before releasing the vectors and the INVALIDATE function made to work.

SET MM ON, SET MM OFF, and SHOW MM commands were added to enable and disable memory management. At the same time DS$MMON and DS$MMOFF were modified to prevent the program from disabling memory management if the operator enabled it.

SYS$SETIMR and SYS$CANTIM were enhanced to change mode to kernel before doing MTPR's or disabling interrupts. This resolved some problems caused by interrupting programs that had set the processor mode to user.

Cleanup was enhanced to reset memory management and re-initialize the SCB after the program cleanup is complete.

Dummy entry points were added to $DS_DSSDEF and the entry vectors to accomodate SYS$LKWSET, SYS$SETAST, SYS$SETPRI, SYS$SETRWM, SYS$ULKPAG, SYS$ULWSET.

SYS$FAO used in stand-alone was modified to fix problems with !*? directives.

The known device data base was updated to include DL11, LP25, RX02, RX211, TM78, TS04, TS11, TU58, and TU78 support. The descriptor for KA780 has changed, the optional arguments are now G-floating, H-floating, WCS last address, and accelerator type.

An error is now gererated if the processor type specified in the system identification register is not 1, indicating an 11/780.

Error text and status codes are returned if the supervisor is run from a command file. The next release will support running in batch.

DS$LOAD has been enhanced to allow programs to read parts of files indicated by the starting block number argument. The name of the console storage device is now always CSan.

The count of loops on subtests has been fixed to correctly reflect the number of passes thru the specified subtest.

The logout for SBI WRITE timeouts now includes the timeout address register.

The ABORT command and DS$ABORT now reset the stacks and reset the processor mode before doing program cleanup.

Diagnostic Supervisor release notes V5.4

Release 5.4 of the diagnostic supervisor has been made. ESSAA.EXE 5.4;214 and ECSAA.EXE 5.4;121 represent 11/780 and 11/750 supervisors. Relevert files can be found on PROTO 3 in Tewksbury as follows:

```
SYS$LIBRARY:DIAG.MLB;611
[DS]ESSAA.EXE;214     11/780 STAR supervisor
[DS]ECSAA.EXE;121     11/750 COMET supervisor
[DS]EVQDB.EXE         QIO driver for RP04,5,6
[DS]EVQDL.EXE         QIO driver for RL02
[DS]EVQDR.EXE         QIO driver for RM80,RP07,RM03,RM05
[DS]EVQDM.EXE         QIO driver for RK06,RK07
[DS]EVQXF.EXE         QIO driver for DR780
[DS]EVQTM.EXE         QIO driver for TM03/TE16,TU77,TU45
[DS]EVQTS.EXE         QIO driver for TS11
[DS]DIAGBOOT.EXE      Secondary Bootstrap
```

Please use these components in testing programs. Needless to say if any problems are encountered please report them to me immediately so their solutions can be included in the next release.

Release notes

- Support for drivers was upgraded to reflect VMS 2.0 interface changes. Loadable drivers have been re-released to reflect Release 2.0 of VMS.

- The supervisor can be run from a VMS command proceedure. When the supervisor is running under VMS it uses RMS to access SYS$INPUT and SYS$OUTPUT.

- Bug fixed to remove supervisor error when supervisor is started again after interrupting a script.

- The APT interface will now generate an error if no PTEXT is found when a process PTEXT command is given.

- A buffer in show device was increased to 256 characters.

- Internal processor registers can be examined with EXAMINE Pxx, where xx is a register number in the current radix. If only the decimal values are available Try EX P%D127 to get register ^X7F. Internal registers can be changed with the DEPOSIT command.

- Channel services was fixed to disable interrupts before releasing the vectors and the INVALIDATE function made to work. Also SETMAP now sets and clears all map registers and allows all to be used for transfers.

- Support was added for the RL02 as a boot device.

- Time conversion routines were changed to fix loss of precision in hundredths.

- SET MM ON, SET MM OFF, and SHOW MM commands were added to enable and disable memory management. At the same

time DS$MMON and DS$MMOFF were modified to prevent the program from disabling memory management if the operator enabled it. Also the return code in R0 now indicates the previous state of memory management and an error return if DS$MMOFF was trying to disable memory management and the operator had enabled it. If memory management is enabled the stack for each mode is write protected against lower access modes.

- DS$ASKSTR now returns 0 in R1 if the default answer was given to a query.

- SYS$SETIMR and SYS$CANTIM were enhanced to change mode to kernel before doing MTPR's or disabling interrupts. This resolved some problems caused by interrupting programs that had set the processor mode to user.

- Cleanup was enhanced tc reset memory management and re-initialize the SCB after the program cleanup is complete.

- Corrected errors induced when more the 2 megabyte of memory was present.

- Dummy entry points were added to $DS_DSSDEF and the entry vectors to accomodate SYS$LKWSET, SYS$SETAST, SYS$SETPRI, SYS$SETRWM, SYS$ULKPAG, SYS$ULWSET.

- SYS$FAO used in stand-alone was modified to fix problems with !*? directives.

- The known device data base was updated to include DL11, LP25, RX02, RX211, TM78, TS04, TS11, TU58, and TU78 support. The descriptor for KA780 has changed, the optional arguments are now G-floating, H-floating, WCS last address, and accelerator type.

- An warning is now generated if the processor type specified in the system identification register is not correct.

- DS$LOAD has been enhanced to allow programs to read parts of files indicated by the starting block number argument. The name of the console storage device is now always CSan.

- The count of loops on subtests has been fixed to correctly reflect the number of passes thru the specified subtest.

- The logout for SBI WRITE timeouts now includes the timeout address register.

- The processor mode and and all stacks and registers are now reset before the users cleanup code is executed.

1.0 BUFFERED I/O SUPPORT

The supervisor now includes support for QIO drivers using buffered I/O (such as line printers).

2.0 HELP FACILITY

The new Supervisor includes a HELP facility. It is invoked via the "HELP" command. In general it is identical functionally with the VMS HELP command. Additionally, it supports gathering HELP on diagnostics via typing the diagnostic code (ES, EV, EC pre-fix followed by diagnostic name, as "EVRAA", etc.) as the first keyword after HELP (e.g., "HELP EVRAA"). In standalone mode the HELP command will function off of any load media (including magtape and console media). In user mode however, it will not function from magtape or the console media, as these are not fully supported by VMS RMS (see below, description of Supervisor RMS).

3.0 LOAD AND SCRIPT

The Supervisor LOAD (which loads files for the commands RUN and LOAD as well as for diagnostics via the $DS_LOAD service call) and SCRIPT file handling services have been modified to use RMS file services.

4.0 MAGTAPE

The supervisor now supports magtape (TM03 only) as load media. TS-11 and TU-78 will be supported in a future release.

5.0 RMS IMPLEMENTATION

The diagnostic supervisor implementation of RMS is similar to the VMS implementation, but is specifically adapted to the standalone DS environment.

DISCLAIMER

Since there are currently no diagnostics which use RMS services, we can not guarantee 100% functionality of RMS code, though we have attempted thorough QA testing. However, services within the Supervisor which use RMS will function correctly.

Console Device Naming

Previous to release 6.0, due to a file name-handling loophole in the file load routine,

it was possible to specify the console device as "CS:". This is not consistent with Supervisor convention, which requires complete device names (i.e., "DMA0:" not "DM:" even if only one RK06/07 is attached). In Version 6.0, "CS:" will not function on COMET (11/730); it will fail when the TU58 driver attempts to access a non-existent drive. The solution is that full device names MUST be typed. In future versions of the Supervisor, this naming "loophole" will be closed, and omitting the controller letter or drive number in the console device specification will return an illegal device error (RMS$_DEV).

5.1 User Mode

To consistently support RMS in user mode (where RMS calls are passed through to VMS), the Supervisor SET LOAD command will function correctly in user mode (as will SHOW LOAD). This is done by translating or reassigning SYS$DISK for device names, and via an RMS housekeeping routine to alter default directory. The Supervisor exit handling routine will restore the original values: however for this to occur, the Supervisor must be terminated either by ^Z or by a ^Y, EXIT sequence. If the (VMS) STOP command is used, the original default directory is not (and CAN not be) restored.

5.2 Differences Between DS RMS And VMS RMS

Supervisor RMS functionality not duplicated by VMS:

- Supervisor RMS supports files on the console media (floppy or TU-58). These files are in RT-11 format, but the Supervisor RMS $GET facility will provide file records as if the file were an ODS CR-attribute, variable length record file. The FAB's RFM field (FAB$B_RFM) will have the value 4.

- Random by RFA (record's file address) access is supported for magtape as well as for conventionally random-access devices. VMS does NOT support magtape random access. In general, random file access should not be used by any level 2 diagnostic, since if the load device is magtape, the function is unsupported in user mode.

VMS RMS functionality not duplicated by Supervisor:

- The Supervisor does not support NAM blocks in any form.

- The Supervisor supports only one form of XAB (the FHC XAB, returning file header characteristics). Any other XAB in the FAB's XAB chain will be ignored, though no error will result. This is because the information is not meaningfull in the diagnostic environment.

- In the FHC XAB, the LRL (longest actual record length) field is not supported. It is meaningful only for ODS files in any case. It will be set to maximum record length always.

- Only sequential files are supported. An attempt to open a file with INDEX or RELATIVE organization results in error RMS$_ORG (illegal file organization).

- Only READ-type functions are supported, since the supervisor does not support write access to any media. To be specific, the functions which the Supervisor RMS package implements is:

  1. $OPEN (access a file on media and initialize FAB)

  2. $CONNECT (Define a record stream for access to a file, initialize RAB)

  3. $GET (Copy a record within the file to a user buffer)

  4. $READ (Unformatted read from file, beginning on block boundary)

  5. $DISCONNECT (Close out a record stream)

  6. $CLOSE (Close out any record streams associated with FAB, and de-access file attached to FAB)

- The bits in the FOP and ROP fields are not significant, except for the BIO bit in the ROP field (specifying block I/O via the $READ call).

- Asynchronous file operations are not supported.

- RMS operations to the terminal are not supported.

- A user may have up to three files open simultaneously.

- Multibuffering and multiblocking are not supported.

- Random access by key or record number is not supported.

- The RAB TMO (time out) field is ignored.

6.0 BUG FIXES

The following Supervisor "bugs" have been fixed for the

6.0 release:

- Channel services incorrectly calculated the number of map registers required for a transfer.
- The operator request routine incorrectly defaulted parameters when prompting was disabled.
- The COMET machine check log-out was incorrect.

1.0 BUG FIXES

- BUG: "EXAMINE PSL" with "/WORD" or "/BYTE" would use only low-order bits of PSL, but the high-order fields would still be decoded and typed as 0.

  FIX: PSL is now unaffected by "/WORD" and "/BYTE"; it is always longword.

- BUG: Previous to Version 6.0, the console storage device could be specified by "CS:". This is at odds with Supervisor convention which requires full device name given in attach command. It results from an accidental "loophole" in that only the first two characters were checked, and an unspecified unit number was converted to 0. In Version 6.0, the unit number conversion was (also accidentally) removed, and using the "CS:" form to Version 6.0 causes a device error on COMET (the STAR console load routine does not use unit number) when the TU58 attempts to access a non-existant (negative) unit.

  FIX: The "CS:" short form is no longer allowed. It will result in an "invalid or illegal device" RMS error (RMS$_DEV) during name parsing. You must type the full device name (e.g., "CSA0:").

2.0 ENHANCEMENTS

- The Supervisor now supports a DIRECTORY command. The format is:

  ```
  DIRECTORY [device:][[directory]]
  ```

  Examples:

  ```
  DS> DIRECTORY DBB3:[SYSMAINT]
  DS> DIR MTA0:
  DS> DIR CSA0:
  DS> DIR [DS]
  DS> DIR DBB3:
  DS> DIR
  ```

  Device and directory default to the current load path (original default is "<load device>:[SYSMAINT]" or last SET LOAD specification). All load media are supported. The format is identical to the standard default VMS directory format. There are no qualifiers. Only file name and version (if applicable, i.e., not on RT-11 media) are typed, and there is no provision for typing dates, sizes, etc.

- To support the AUTOSIZER program, the supervisor's ATTACH command processor has been made externally callable. The entry point is DS$ATTACH

(cmdline,[prompt]) where cmdline is the address of a string descriptor for the ATTACH command to be parsed, and prompt is the address of a descriptor for a buffer (MUST be at least 1 byte long if prompt is given!) to receive the ASCIC text with which to prompt the operator if the command line is incomplete or incorrect (on error, the prompt string is set, and the cmdline descriptor length is decreased to include only the correct part of the input line). The prompt argument is optional and in general is useless except within the Supervisor.

Format:

```
$DS_ATTACH (cmd, [pmt])          ! Bliss
$DS_ATTACH_x    cmd, [pmt]       ; Macro
```

- The Supervisor HELP routine has been made callable, to allow programs implementing a conversation mode to include a HELP command. You pass the address of a descriptor of a command string; this string is just what you'd type on a Supervisor HELP command line, except leave out the "HELP" keyword. You can access the program's usual HELP file (preferable) by prefixing whatever keys the operator types with the program name. Keep in mind that HELP will fiddle with the adapter and controller, etc. in the path to the load device.

  Format:

```
$DS_HELP (keylst)                ! Bliss
$DS_HELP_x      keylst           ; Macro
```

- User control of ^C

  - In some diagnostics, it may be necessary to call DS services which do implicit "$DS_BREAK" calls during sections of code which must not be interrupted by "^C". To solve this problem, new functionality has been added to the $DS_CNTRLC service. A second argument, DISABL, has been implemented. The value of this argument defaults to "0", and the DS$CNTRLC routine accepts argment lists of only one argument to maintain compatibility with currently released diagnostics. If the low bit (bit 0) of DISABL is SET, ^C will be ignored by the Supervisor. If a user ^C handler is set, it will not be called, nor will the Supervisor itself take any special action beyond setting it's ^C flag bit. Note that if ^C recognition is disabled, and a ^C character is typed, the ^C will be honored as soon as ^C recognition is re-enabled. Therefore, the ^C is effectively postponed beyond the non-interruptable segment of code, but is not lost.

    Format:

```
$DS_CNTRLC ([astadr], [disabl]) ! Bliss
$DS_CNTRLC      [astadr], [disabl]      ; Macro
```

. Previously, the user ^C handler routine had no value. Now, the user handler has the option of telling the Supervisor to handle the ^C normally even though there was a user handler defined. To do this, the user's handler should return a failure status code (R0 bit 0 clear). If the user handler returns a success code, the Supervisor will do nothing further.

- It is possible for messages to be typed out asynchronously, by AST routines for example. This type-out may occur while the operator is keying in commands and/or data. In such a circumstance, VMS will retype the input line automatically (as if a ^R had been typed), but the Supervisor would just let the line "dangle". Now, the Supervisor will perform the "^R"

3.0 NEW LOAD DEVICE SUPPORT

- Supervisor RMS (and therefore all Supervisor file utilities including HELP, DIRECTORY, DS$LOAD, RUN, and LOAD) now will function with the TS-11/TS-04 tape drive sub-system. This addition is transparent to users of the utilities, syntax and programming-wise.

- A user control-C handler returning failure status under APT would cause the Control-C to be ignored, since status information was cleared before the user routine was CALLed. This has been fixed by deferring the status clear.

- DIRECTORY - due to an inconsistency in string descriptor handling, access violations could occur during DIRECTORY depending on previous stack contents. Specifically, if a HELP command with keywords was entered previously, DIRECTORY would fail, but would succeed if HELP without any keywords was typed. The problem has been fixed.

NOTE

Addition to directions for DIRECTORY command:

1) In order to get a directory in user mode, the target device must be mounted (do NOT use the /FOREIGN qualifier!). You can not get a directory from the console device in user mode, since VMS RMS does not support it.

2) The target device must be attached before you can get a directory of it. This applies in both standalone AND user modes. This is due to the fact that the Supervisor does not enforce device naming conventions, and therefore requires access to a Ptable to figure out what type of directory structure it is accessing.

3) The directory command will not work on ODS structure level 1 disks in standalone. (This is a bug, and will be corrected in a future version of the Supervisor!)

1.0 BUG FIXES

- o If the COMET supervisor is loaded from a Massbus device, it currently attaches the load MBA as RH0, irregardless of the actual unit number. In version 6.3, this has been fixed, and it will attach the correct unit number.

- o In the $DS_DEVTYP macro, under certain circumstances the supervisor could be fooled into thinking that a device name string was part of a Ptable descriptor when it actually was only a name. This is because it looks 5 bytes beond the end of the string for a byte 80(x), and if the byte IS 80, the supervisor thinks it is a Ptable descriptor. This has been fixed by adding a null byte to the end of the DEVTYP string list. This does not affect any diagnostic already assembled as it is transparent to both diagnostic and supervisor, occuring only within the library DIAG.

- o For the $DS_ABORT macro, there was a difference in syntax between the MACRO and BLISS forms, as BLISS required 'PROGRAM' while MACRO would accept PROGRAM or program. BLISS will now accept PROGRAM and program, as well as 'PROGRAM' for compatibility with older versions.

- o The $DS_WAITMS macro has been fixed so it will work correctly.

- o The DV11 is now ATTACH-able. Previously the Supervisor knew of the device but would not allow ATTACH-ing it.

- o RMS fixes (to match VMS RMS functionality more closely):

  - When $GET returns RMS$_RTB, the RAB$L_STV field will contain the full record size.

  - $READ will return the block number read in the first longword of the RFA (RAB$L_RFA0). The byte offset of the RFA (RFA$W_RFA4) is 0.

- o SBI silo register dump (11/780 only). Due to an incorrect assumption as to instruction context in indexed addressing mode, the wrong field was decoded for function ID. This problem has been corrected.

- o A fix to DS$WAITUS to promote more correct and consistent behaviour. This involves ensuring that the interrupt and error bits in the timer CSR are cleared.

- o It is now legal for a section name to include the characters "$" and "_". Previously, they could be used in a diagnostic section name definition, but the Supervisor /SECTION qualifier (RUN and START commands) would not recognize them as legal. The Supervisor will now accept these characters. In conjunction with this, the function code CLI$K_SYMBOL has been added to the $DS_CLIDEF macro, and may be used by any diagnostic. This code causes PARSE to recognize a string composed of the characters "A" to "Z", "a" to "z", "0" to "9", "$", and "_".

- o In standalone, the $SETPRT service would return an incorrect

address vector. The start address would be 1FF hex, due to a reversal of operands in a BICL3 instruction. The address will now be correct.

o In standalone, the $DS_RELBUF (or $CNTREG) would fail with an ERRSUP (Supervisor fatal error) if the page geing released was in P1 or SYS space and had been modified with memory management on (M bit set in PTE). V6.3 corrects this problem.

2.0 ENHANCEMENTS

o Several QIO driver service entry points have been added to support level 2R to level 2 conversion: IOC$ALLOSPT, EXE$MAXACMODE, and EXE$READLOCK. These entry points are defined in the DIAG.MLB $DS_QIOVEC macro.

o Previous to V6.3, it was possible to attach a device to any of the predefined "main buses" recognised by the supervisor. Now, an error message will result if the main bus name does not match the CPU type; for example, if a device is attached to an SBI on a COMET processor the error message is "there is no SBI on this type processor." Also, two new "main bus" names have been defined. These can be used on any VAX implementation. They are PHAD (for "Physical Address Decoder") and HUB.

o The $DS_SETVEC macro (DS$SETVEC routine) now allows setting the processor mode in which an interrupt will be handled, without changing the handler address. To do this, set the SRVADR argument to 0, and use the CODE argument as usual. Bits 2 to 31 of the SCB vector will be unchanged, but bits 0 and 1 will still be set to CODE.

o New commands. These support customer service and manufacturing in using terminals narrower than 132 columns, without losing data.

- SET WIDTH n
- SHOW WIDTH

These commands will be accepted and carried out in user mode, but will have no effect on terminal characteristics (use VMS SET TERMINAL/WIDTH=n before running Supervisor). In standalone, a newline is forced when the cursor (printhead) passes column n.

o In user mode (only), the Supervisor will accept subdirectories in file specifications.

o HELP FILE: Where possible, I have included standard (fixed) CSR and vector addresses in the DEVICE portion of EVSAA.HLP. Also, the lines which used to read "Device:" now read "Description:" to avoid possible confusion over actual names. Finally, the recommended generic names have been brought into agreement with the "MASTER LIST OF PDP-11/VAX-11 DEVICE NAME CONVENTIONS" maintained by Kerby Altmann of VMS Development.

o Beginning with V6.3, the Supervisor .DOC file will contain a concatination of all release note files (from V5.3 on). This will allow easier reference to warnings on use of Supervisor features, as well as tracking enhancements/bug fixes.

1.0 BUG FIXES

- QIO driver entry points IOC$ALLOSPT, EXE$MAXACMODE, and EXE$READLOCK, which were supposed to have appeared in version 6.3, somehow didn't. They are present in 6.4.

- ECSAA attached MASSbus load device incorrectly. This has been fixed.

- Most bits of the 11/750 UBI register are undefined, specified as MBZ (Must Be Zero) but are actually unpredictable. This is confusing in the output of a $DS_SHOWCHAN call. Therefore, the Supervisor will now mask out these undefined bits.

- The $SETPRT system service (standalone) would not work if the caller was not already in Kernel mode. Version 6.4 will do a CHMK to insure kernel mode privilege for the routine.

- The byte context PRVPRT output argument of $SETPRT was being written as a longword. Version 6.4 will write it as a byte.

- The diagnostic cleanup code was not called when a loop on subtest terminated. This has been fixed.

- If a range of tests were used with the /SUBTEST qualifier, the diagnostic would loop on the first subtest with the specified number, irregardless of which test it was in. Now, it will only loop on the correctly numbered subtest in the last test number given.

- The VMS console device non-interrupt I/O drivers have been incorporated into the Diagnostic Supervisor. This means console reads which fail due to media bad spots will be re-tried. (This is more of an enhancement, but is listed under bugs because Manufacturing has complained of trouble with console bad spots under APT control).

- The bliss diagnostic library (DIAG.L32) had several bugs in it, including incorrect macros for $DS_$INITIALIZE and $DS_$STRING, many errors in %ERROR statements to point out bad macro useage, and incorrect compilation of $DS_PRINTSIG macro. These have all been fixed.

- A bug in $DS_SETVEC prevented setting the mode in which a 'soft-vectored' exception was handled. Soft vectored exceptions include BPT and T-bit traps. This problem has been fixed.

2.0 ENHANCEMENTS

- FIELD definitions for all Ptable types known to the Supervisor have been added to the Bliss library (DIAG.L32). They can be used via the $DS_HPODEF macro; for example, "$DS_HPODEF ($DS_DZ11_DEF)" allows

access to the basic Ptable fields, plus the fields of the DZ11 Ptable. Also, literals representing the length of these Ptables have been added. There are no FIELD definitions for Ptables which have no device dependent fields.

- Correct spelling of many Supervisor messages, and change to mixed case for greater legibility.

3.0 MISCILLANEOUS

- Note: The $SETAST system service (as documented in the VMS System Service manual) is supported by the Supervisor in standalone. It was added in version 5.4 for Supervisor internal use and was not documented. It may be used by level 2 and 3 programs freely.

- Note: The Diagnostic Design Guide is in error over the PRINT ($DS_PRINTX, $DS_PRINTB, $DS_PRINTF) format argument. It should be counted ascii (ASCIC) not ASCID.

3.1 Previously Undocumented Gems Of Wisdom: Linking Diagnostics

- Bliss-32 => If a diagnostic is written entirely in Bliss-32, without any Macro-32 modules, the diagnostic must be linked with a DS.STB file defining Diagnostic Supervisor entry points (for system services, etc.).

  DS.STB can be created easily from the following Macro-32 source:

```
        .TITLE  DS.STB  Symbol table to link with BLISS modules
        .LIBRARY        'SYS$LIBRARY:DIAG'
        $DS_DSADEF      GLOBAL
        $DS_DSSDEF      GLOBAL
        .END
```

  And assembled/linked as follows:

```
        $ MACRO DS
        $ LINK/SYMBOL_TABLE=DS/NOEXECUTABLE DS
```

  Link <diag> as such:

```
        $ LINK <qualifiers> <diag>+DS.STB
```

- Macro-32 => The $DS_BGNMOD and $DS_ENDMOD macros use a label called $M0 for checking. The $DS_BGNMOD macro defines "$M0 == 0", and the $DS_ENDMOD defines "$M0 == $M0 + 1". Therefore, a missing $DS_BGNMOD macro will cause an "undefined symbol" error at assembly time in that

module, and a missing $DS_ENDMOD will cause a "multiply defined global" error at link time, due to the fact that correct module(s) define $MO as 1 and modules missing $DS_ENDMOD macros define $MO as 0.

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.5
4-DEC-1981

A. GENERAL ENCHANCEMENTS

1. UNIBUS Mapping

The entire UNIBUS is now mapped, instead of just the I/O pages. This allows diagnostics to get UNIBUS NXM errors instead of access violation errors.

2. Channel Services status

By popular demand, channel services has been enhanced to return additional information:

- o UNIBUS NXM errors reported by bit CHS$M_BUSNXM
- o MASSbus writecheck errors reported by three bits:
  - a. CHS$M_CHNDPE is set if either of the other two are set:
  - b. CHS$M_MBAWCKLWR is set if the MASSbus write check lower bit is set.
  - c. CHS$M_MBAWCKUPR is set if the MASSbus write check upper bit is set.

3. APT troubleshooting

To enable troubleshooting of diagnostic problems under APT, the hardware context will not be initialized when an ABORT command code is received from APT, if errors have occurred.

4. Automatic diagnosis assistance

There are several programs (including APT, RD, and EVXBB) which act as operators to VDS. Currently, this process requires parsing of the VDS ASCII output. With Version 6.5, VDS is growing to make the task of automated diagnosis easier: A new flag has been implemented, called BINARY. When this flag is set, VDS will output a binary 'type byte' as the first byte of every message (including prompts and diagnostic PRINTS). This type byte will indicate the message class (e.g., command error, program start, harderror, etc.) The type codes are defined in macro $DS_TYPECODE in DIAG.MLB and DIAG.L32. Currently, the implementation is incomplete. Most messages for this release will have a type code of 0 (ds$k_type_general). Only prompts, program sequence messages (start, first_pass, end, etc.), and command messages will generate meaningfull type codes. The BINARY flag is off by default, and is not turned on by SET FLAGS ALL (or turned off by CLEAR FLAGS ALL).

5. DEVICE UNIQUENESS

The concept of a unique device has been altered. Previously two devices with the same name were considered the same, and could not be ATTACHed at the same time. To support network nodes

(which may be accessible through several different network adapters), TWO DEVICES WITH THE SAME NAME BUT DIFFERENT LINKS ARE NOW CONSIDERED DIFFERENT. This means that a device attached to the wrong adapter can not be 'edited' by re-attaching it. The new DEATTACH command gives the ability to correct any ATTACH errors.

6. Error messages

All error messages (ERRHARD, ERRSOFT, ERRDEV, ERRSYS, and "Software detected error") now include a trailing "End of error" line heralded by "********". This makes it easier to separate errors and assorted intervening text on printouts. It was particularly devised in conjunction with the BINARY flag (see item 4 in this section), and has its own unique type code.

B. COMMAND ENHANCEMENTS/ADDITIONS

1. New Qualifier /BRIEF

This qualifier can be used on the SHOW DEVICE and SHOW SELECT commands. It will cause only the device name and type fields to be printed.

2. New qualifier /ADAPTER=name

This qualifier can be used on the DEATTACH, SELECT, DESELECT, and SHOW DEVICE commands. It will limit the command to devices attached to the adapter with device name 'name'. E.g., for select, "SELECT/ADAPTER=DW0 ALL" will cause all devices attached to DW0 (but not devices attached in turn to those devices) to be selected for test. The special adapter name "HUB" is recognised, but not "SBI", "CMI", etc.

3. DEATTACH

This command reverses the ATTACH command for a device. It can be used to remove incorrectly attached devices. The /ADAPTER=name qualifier is REQUIRED on this command. Since Ptables are linked in a tree structure, it makes no sense to de-attach a device which has other devices attached to it, unless those devices are also de-attached. Therefore, the DEATTACH command will recursively "seek and destroy" all lower level devices for each device it de-attaches. E.g., if DW0 has devices DMA and TTA attached to it, TTA has device TTA0 attached to it and DMA has DMA0 attached to it, typing "DEATTACH/ADAPTER=HUB DW0" will cause DW0, DMA, TTA, DMA0, and TTA0 to be de-attached. a message will be typed for each as it is de-attached. (note that the message order may appear strange, since the de-attach routine does a post-order search).

4. EXIT

In on-line mode, this command will execute a VMS $EXIT system service to terminate the VDS image. In standalone, it will do a HALT after re-initializing the machine (a console CONTINUE will return to VDS command mode, however).

5. SHOW BASE

Displays the current base value. (as in "SET BASE").

6. SHOW DEFAULT

   Displays the current default radix and data size.

7. SHOW SECTIONS

   Lists the SECTION names supported by the current diagnostic. An error results if no diagnostic is loaded.

8. SHOW STATUS

   Prints the name, revision, current time, error count, section, pass count, test, subtest, and PC for a CNTRL/C'd diagnostic. If there is not a running diagnostic, an error is typed.

9. SHOW SUPPORT

   Prints the list of devices supported by the current diagnostic. An error results if no diagnostic is loaded.

C. GENERAL CLI/PARSE ENHANCEMENTS

1. New PARSE features

   Special function codes have been added to PARSE to improve command parsing for the VDS CLI (they can also be used by diagnostics):

   o CLI$K_EOL: Accepts a string consisting of blanks (any number of spaces and/or tabs; including none of either) followed by either end of line (no more characters in string) or a "!" character (comment delimiter). It will call the action routine and branch through the miss label, as appropriate.

   o CLI$K_SLASH: Accepts blanks, followed by the character "/" followed by blanks.

   o CLI$L_COMMA: Like CLI$K_SLASH, but expects a "," instead of a "/".

   o CLI$K_VALUE: like CLI$K_SLASH, but expects either an "=" or a ":" instead of a "/". "=" and ":" are exactly equivalent (as in VMS DCL).

2. CLI improvements:

   o CLI uses the features listed above to make command syntax more flexible. For example, the qualifier "/PASSES:1" is the same as " / PASSES = 1".

   o The old CLI had two error messages: "Illegal command" and "Ambiguous command"; both of which were usually rather ambiguous themselves. The latter has been replaced by "Incomplete command" and the rules for generating each have been tightened:

     a. A command is illegal if something unexpected is found. E.g., the command "XYZZY".

b. A command is incomplete if something expected is NOT found. E.g., "CLEAR" lacking any arguments.

Furthermore, both error messages include some information on what was in error. "Illegal command" displays the command line UP THROUGH THE FIRST ILLEGAL CHARACTER. "Incomplete command" displays up to where the expected data should have been. E.g., the command "XYZZY" results in:

```
?? Illegal command
   \X\
```

while the command "CLEAR" results in:

```
?? Incomplete command line
   \CLEAR\
```

D. PTABLE DESCRIPTOR ENHANCEMENTS

1. New directives

o Previously, to request logical (yes/no) data, it was necessary to use the $DS_$STRING directive, supplying prompt, "NO", and "YES" strings. I have implemented the new directive $DS_$LOGICAL. It's only argument is the prompt string. This saves program space, and also allows ABBREVIATION of the words "Yes" and "No". All Ptable descriptors within VDS which require "Yes/No" input have been modified to use this directive.

o Value translation can now be accomplished by the directive $DS_$CASE. The argument is a list of "case_pairs"s. The current VALUE register is compared to the first value of each pair until a match is found; the second value of that pair becomes the new VALUE, and the case directive is exited. VALUE is not altered if no match is found.

Bliss-32:

```
$ds_$case (<1,%x'FFF'>, <2,%x'1CE0'>)
```

MACRO-32

```
$ds_$case <<1,<^xFFF>><2,<^x1CE0>>>
```

E. MISCELLANEOUS ENHANCEMENTS

1. VERIFY

VDS now supports a mechanism similar to the VMS "SET [NO]VERIFY". The flag VERIFY can be set or cleared via "SET FLAG VERIFY" or "CLEAR FLAG VERIFY". When the flag is set, commands read from VDS scripts are echoed to the console. When the flag is clear, command lines are not echoed. This does not affect error or information messages, and has no effect unless commands are being processed from a VDS SCRIPT file : commands read from a VMS command file are considered by VDS to be coming from the console!

!!The flag is CLEAR by default!!

2. MIXED CASE MESSAGES

Many messages throughout VDS have been altered from all caps to mixed case. We feel that this is much easier to read, as well as appearing "warmer", in the line of "human engineering".

F. BUG FIXES

1. Ptable descriptor location

A bug has existed which could cause errors when a program's DEVTYP list existed but had 0 elements. This has been fixed.

2. ATTACH

It is now possible to CNTRL/C out of an ATTACH prompt. Furthermore, ATTACH will not attempt to prompt a SCRIPT or APT for a missing or invalid parameter: the command will be aborted. From a SCRIPT, the error "Prompt not found" is generated. From APT mode, a "Software detected error" is generated.

3. Fixes to DIAG.L32 macros

o $DS_ENDMOD malfunctioned when a module did not contain tests (incorrectly setting an overlayed PSECT which at runtime should contain highest test number). This has been fixed.

o $DS_HPO_DECL has been fixed to correctly accept multiple arguments.

o $DS_$STRING and $DS_$INITIALIZE did not function properly. Now they will.

4. Event flag services

All standalone event flag services have been re-written to correspond to the VMS services. The $SETEF and $CLREF services will return SS$_WASSET or SS$_WASCLR when they should, as will $READEF.

G AUTOMATED DIAGNOSTIC QUALITY ASSURANCE

G.1 Background

Enhancements have been made to the Diagnostic Supervisor to enable it to automatically perform Quality Assurance checks on diagnostic programs.

This version of the Supervisor performs the Normal Start check (Section 3.1 in the "VAX Diagnostic Quality Assurance Checklist"), the Multiple Pass check (Section 3.2), the Loop on Test check (Section 3.7), and the Run Backwards check (Section 3.5). Other sections in the checklist will be added to the Supervisor as time and resources permit.

The Normal Start check does the following: saves the current setting of the Supervisor flags; turns on the TRACE flag; turns

off the following flags: BELL, IE1, IE2, IE3, IES, LOOP, OPER, PROMPT, QUICK, and SEARCH; performs a normal start of the diagnostic program, running all the tests in the specified section; turns the TRACE flag off; and either stops when a QA error is detected or proceeds to the next check.

The types of errors that the Normal Start check will detect are asking for operator input when the no-default flag is set (in the $DS_ASK macro) and reporting a diagnostic error via one of the $DS_ERR macros. When a QA error is detected, the Supervisor will print out information that should help the diagnostic engineer determine why the program failed QA. Then, an overall error summary table is printed, the Supervisor flags are restored to their previous settings, and the diagnostic is aborted.

If the Normal Start check succeeds, the Multiple Pass check is performed. This check performs a number of passes of the diagnostic program. The current setting of the QAMULTIPLEPASS flag is the number of passes that will be executed (see Section G.4). The types of QA errors that can occur are the same as those in the Normal Start check.

If the Multiple Pass check succeeds, the Loop on Test check is executed. This check runs each test QATESTLOOPS times (see Section G.4). Assuming a diagnostic has tests 1, 2, and 3, and QATESTLOOPS has been set to 4, this check will run the initialization code, followed by tests 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, and 3. The types of QA errors that can occur are the same as those in the Normal Start check.

If the Loop on Test check succeeds, the Run Backwards test is executed. This check runs the tests in reverse order. If a diagnostic contained tests 1, 2, 3, and 4, this check would execute the initialization code followed by test 4, test 3, test 2, and then test 1. The types of QA errors that can occur are the same as those in the Normal Start check.

If the Run Backwards check succeeds, the overall error summary table is printed, a line is printed saying that the program passed Auto-QA, the Supervisor flags are restored to their previous settings, and the diagnostic is aborted.

G.2 START/QA

A new qualifier has been added to the START command. If the START command contains the /QA qualifier, QA will be performed on the currently loaded diagnostic program. If the /QA qualifier is present, the /PASSES, /SUBTEST, and the /TEST qualifiers are ignored. The /SECTION qualifier may be used to run QA on a particular section in the program.

G.3 RUN/QA

The /QA qualifier has also been added to the RUN command. It functions exactly the same as the /QA qualifier on the START command.

G.4 QA Flags

There are five new Supervisor flags that will affect QA. However, only two of these are used in the current implementation.

The QAMULTIPLEPASS flag controls how many passes are performed in the Multiple Pass check. This flag may be set as follows:

```
DS> SET QAMULTIPLEPASS 10
DS> SET QAM 3
```

In addition, the current setting of the flag may be printed as follows:

```
DS> SHOW QAMULTIPLEPASS
DS> SHOW QAM
```

Note that the QAMULTIPLEPASS flag may be abbreviated to QAM. The initial (and default) setting for this flag is 10. That is, 10 passes of the diagnostic will be executed in the Multiple Pass check.

The QATESTLOOPS flag controls now many times each test is executed in the Loop on Test check. For example, if a diagnostic program contains three tests and QATESTLOOPS equals 2, this check will first execute the initialization code, then tests 1, 1, 2, 2, 3, and 3.

This flag may be SET and SHOWn in the same manner as the QAMULTIPLEPASS flag. This flag may be abbreviated to QAT. The initial (and default) setting for this flag is 100.

Also, the following command will set all the QA flags to their default settings:

```
DS> SET QADEFAULTS
```

QADEFAULTS may be abbreviated to QAD. To show the default settings (not the current settings) of the five QA flags, the following may be used:

```
DS> SHOW QADEFAULTS
```

G.5 Modifications Necessary For QA

Diagnostic programs wishing to undergo QA must call the $DS_ENDPASS macro in the initialization code. This is necessary for the internal Supervisor sequencing of QA checks. The branch macros discussed in the "Design Specification for Automatic Quality Assurance for Diagnostic Programs" need not be used in the diagnostics for this version of QA.

Because of the way the Supervisor works, the code executed in the diagnostic's initialization code when pass0 is true (in conjunction with the $DS_BPASS0 or $DS_BNPASS0 macros) will be done only once. It is done at the start of the Normal Start

check. Also, the cleanup section will be done only once. It is done at the conclusion of the Normal Start check (before the Multiple Pass check is started).

Thus, the whole QA sequence of checks may be thought of as executing the diagnostic a number-of-passes times, with different things happening in different passes. That is, one pass is executed for the Normal Start check, then QAMULTIPLEPASS passes are executed in the Multiple Pass check, then one pass is executed in the Loop on Test check (with the tests executing more than once), and then one pass is executed for the Run Backwards check (with the tests being run in reverse order).

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.7
03-May-1982

A. GENERAL ENCHANCEMENTS/NOTES

1. This release is almost identical to the Version 6.6 VDS which was released in the previous cycle. However, due to technical difficulties, particularly due to the support of VMS V2-style QIO interface in standalone, while linking VDS under V3 of VMS, it was not possible to release version 6.6 VDS for 11/780 and 11/750 processors.

   This release (6.7) includes all three processors; ENSAA (11/730), ECSAA (11/750), and ESSAA (11/780).

2. VMS V3a support

   Support of VMS V3a includes support for the new system directory structure, where [SYS0.SYSMAINT] replaces the old [SYSMAINT] directory. Earlier versions of VDS will not operate correctly with a 'rooted' SYSMAINT directory. This support requires DIAGBOOT.EXE V4, which was released with the previous release cycle.

   A side-effect of the support of rooted systems is that VDS now fully supports subdirectories in standalone mode.

   NOTE that while VDS 6.7 supports the directory structure of VMS V3a, it is linked under VMS V2.5, and so will run under VMS V2.5 or V3a.

3. New Device support

   VDS V6.7 includes support for the UDA-50 UNIBUS to SI disk adapter/controller, and for the RA- series disk drives which can attach to it.

B. COMMAND ENHANCEMENTS/ADDITIONS/FIXES

1. SELECT/DESELECT

   In version 6.5, the SELECT and DESELECT commands did not handle the new definitions of 'device uniqueness' correctly. If the device name specified was attached to more than one adapter, the commands would simply act upon the first such device found (unless the /Adapter qualifier was used). Since the SELECT command affects the order in which device names are searched, this often meant that it was impossible to DESELECT the device which had been SELECTed; yet no error message was given. Now, such a situation will result in an error message saying that the /Adapter qualifier must be used.

2. HELP

   In the process of major restructuring of the VDS command parser for version 6.5, a bug was inserted in the parsing of the HELP

command. Specifically, any command line which began with 'H' was taken as a HELP command; for example, a 'SHOW' command mis-typed as 'HSOW' was interpreted as if 'HELP SOW' had been typed. This has been fixed for VDS 6.6.

3. SHOW MEMORY

This new command displays the layout of VDS memory. In default mode, the start and end addresses of the diagnostic, VDS, and dynamic memory (used by VDS) are typed, as well as the physical memory size of the system, in both HEX and DECIMAL radix. There are several qualifiers to this command. Note that you can not use these qualifiers prior to the MEMORY keyword. E.g., 'SHOW MEMORY/ALL' but not 'SHOW/ALL MEMORY'.

a. /MAP : same as default mode

b. /DATA_STRUCTURE : shows internal VDS data structure addresses, including all stacks, page tables, SCB, and various simulated VMS data structures (PCB, PHB, etc.). knowing the addresses of these data structures can be usefull in debugging. Note that the memory layout in on-line mode is quite different than in standalone; in on-line mode, none of the aforementioned data structures are present within VDS memory space, and will not be displayed.

c. /BUFFER : Show available space for diagnostic buffers, both above and below VDS (this command is not meaningfull when no diagnostic is loaded). Note that since on-line mode allocates buffer space above VDS via $EXPREG as it is requested, /BUFFER on-line will not show space above VDS.

c. /ALL : Combine /Map, /Data_Structure, and /Buffer

4. DIRECTORY

The DIRECTORY command has been enhanced. Previously, it would accept as arguments only the device and directory parts of a filespec, and would take only one level of directory in standalone mode. Now, you can specify the filename, type, and version fields of the filespec; additionally, both the filename and type fields may include both "*" and "%" wildcards (with the same meaning as under VMS). Furthermore, subdirectory specification is now legal in standalone as well as on-line.

5. SUMMARY

The SUMMARY command has been enhanced. Whether or not the diagnostic program contains summary code, VDS will generate a header containing the name of the diagnostic and a summary of all HARD, SOFT, DEVICE, SYSTEM, and SUPERVISOR DETECTED errors which occurred during the run.

C. AUTOMATED DIAGNOSTIC QUALITY ASSURANCE

1. Introduction

These sections will explain what functionality I have added

to the Automatic Quality Assurance enhancement to the Diagnostic Supervisor, and what problems I have fixed since the 6.5 version of the Supervisor. Note that when I speak of versions of QA or versions of the Diagnostic Supervisor, I am actually speaking of the same thing.

2. Additional Functionality

The following gives the two new QA checks that have been added to the 6.6 version:

a. Loop On Subtest check:
This check performs a loop on every subtest in the diagnostic. The number of loops is user-controlled via the QASUBTESTLOOPS Supervisor flag. The following is an example of how this check works:

Assumptions:
Diagnostic EXXXX contains two tests.
Test one has three subtests.
Test two has two subtests.
QASUBTESTLOOPS has been SET to 3.

Order of Execution (Test#.Subtest#):
1.1
1.1
1.1

1.1
1.2
1.2
1.2

1.1
1.2
1.3
1.3
1.3

1.1
1.2
1.3
2.1
2.1
2.1

2.1
2.2
2.2
2.2

2.1
2.2
End of Pass

In the above, the blank lines represent places where QA internally aborts the diagnostic and restarts it so as to start looping on the next subtest. The ENDPASS routine is only executed at the very end of the check. However, the initialization code and the cleanup code are both

executed once for each of the above groupings.

b. Error Phase One check:
This QA check forces as many error reports as possible to be printed. This is a way of insuring that the parameters passed to the error routines are correctly coded, as well as allowing the user to check the format of the error messages. The following lists some points that must be considered when writing a diagnostic that will be QA'ed using Auto-QA:

i. Any error call that was not forced (i.e., was not preceeded by a branch macro in which QA forced the branch to change sense) will be treated as a QA error, resulting in termination of Auto-QA.

ii. No two branch macros should occur in sequence without an intervening error macro call. That is, the sequence should always be one branch macro followed by one error macro call.

iii. No branch macros should occur in a subroutine. Branch macros in subroutines may cause QA to go into an infinite loop and print error reports ad infinitum! This may be fixed in a later version of QA.

iv. The branch macros for use in MACRO diagnostic modules are listed below. For an explanation of the parameters, see Revision 1.1 of the "Functional Specification for Automatic Quality Assurance for Diagnostic Programs" memo. Contact Jack Stansbury (TW/F18) for a copy of this memo.

```
$DS_BLSS        ADR, NUM
$DS_BLSSU       ADR, NUM
$DS_BLEQ        ADR, NUM
$DS_BLEQU       ADR, NUM
$DS_BEQL        ADR, NUM
$DS_BEQLU       ADR, NUM
$DS_BGEQ        ADR, NUM
$DS_BGEQU       ADR, NUM
$DS_BGTR        ADR, NUM
$DS_BGTRU       ADR, NUM
$DS_BGTRU       ADR, NUM
$DS_BNEQ        ADR, NUM
$DS_BNEQU       ADR, NUM
$DS_BVS         ADR, NUM
$DS_BVC         ADR, NUM
$DS_BCS         ADR, NUM
$DS_BCC         ADR, NUM
$DS_BBS         POS, BASE, ADR, NUM
$DS_BBC         POS, BASE, ADR, NUM
$DS_BBSS        POS, BASE, ADR, NUM
$DS_BBSC        POS, BASE, ADR, NUM
$DS_BBCS        POS, BASE, ADR, NUM
$DS_BBCC        POS, BASE, ADR, NUM
$DS_BBCCI       POS, BASE, ADR, NUM
$DS_BBSSI       POS, BASE, ADR, NUM
$DS_BLBS        SRC, ADR, NUM
```

```
$DS_BLBC        SRC, ADR, NUM
$DS_BERROR      ADR, NUM
$DS_BNERROR     ADR, NUM
```

v. The branch macros for use in BLISS diagnostic modules are as follows (note that the BCS, BCC, BVS, and BVC branch macros are not implementable in BLISS):

```
$DS_BLSS    (VAL, NUM)
$DS_BLSSU   (VAL, NUM)
$DS_BLEQ    (VAL, NUM)
$DS_BLEQU   (VAL, NUM)
$DS_BEQL    (VAL, NUM)
$DS_BEQLU   (VAL, NUM)
$DS_BGEQ    (VAL, NUM)
$DS_BGEQU   (VAL, NUM)
$DS_BGTR    (VAL, NUM)
$DS_BGTRU   (VAL, NUM)
$DS_BGTRU   (VAL, NUM)
$DS_BNEQ    (VAL, NUM)
$DS_BNEQU   (VAL, NUM)
$DS_BBS     (POS, BASE, NUM)
$DS_BBC     (POS, BASE, NUM)
$DS_BBSS    (POS, BASE, NUM)
$DS_BBSC    (POS, BASE, NUM)
$DS_BBCS    (POS, BASE, NUM)
$DS_BBCC    (POS, BASE, NUM)
$DS_BBCCI   (POS, BASE, NUM)
$DS_BBSSI   (POS, BASE, NUM)
$DS_BLBS    (SRC, NUM)
$DS_BLBC    (SRC, NUM)
$DS_BERROR  (VAL, NUM)
$DS_BNERROR (VAL, NUM)
```

After all the errors are forced, a table is printed that summarizes the error numbers. This is the Forced-Error Summary table. This table looks like the following:

Summary of Forced-Errors

```
Test  X1   Subtest  Y1
--------   ----------
aaaaa-bbbb ccccc-dddd ...
mmmmm-nnnn ...

Test  X2   Subtest  Y2
--------   ----------
AAAAA-BBBB CCCCC-DDDD EEEEE-FFFF GGGGG-HHHH ...
MMMMM-NNNN OOOOO-PPPP QQQQQ-RRRR ...
...
```

The error numbers are printed in a test/subtest order, with the error numbers sorted within each test/subtest.

The Xi and Yi are the test and subtest numbers,

respectively. The 'aaaa', 'cccc', ..., 'AAAA', 'CCCC', etc. are the error numbers that are passed to the $DS_ERRxxxx macro.

The 'bbbb', 'dddd', ..., 'BBBB', 'DDDD', etc. are the number of times that error was forced. This number should be one except in those cases where the same error number is used more than once in one subtest.

If the diagnostic runs on a subset of the total number of tests in the DEFAULT section (by using the /TEST qualifier), the Overall Error Summary table will not be printed.

However, if the /TEST qualifier is not used, the Overall Error Summary table will be printed. This table gives the number of errors that occurred in each QA check. For the 6.6 version of the Supervisor, these numbers should be either all zeroes (in which case a line is printed saying the program passed Auto-QA), or else there will be at most one error (since Auto-QA is aborted upon detection of a single QA error).

3. Auto-QA Problems in the 6.5 Version

The following is a list of problems with QA that were discovered in the 6.5 version of the Supervisor:

a. The Halt-On-Error flag was not cleared before starting QA. I had meant to clear all the Supervisor flags when QA started, and then restore them when QA ended. However, I forgot this one flag. Although it probably will not cause any problems in the 6.5 version, it would have caused problems in this new version.

b. If a previously-started diagnostic is running, a Control-C is typed, and then a CONTINUE Supervisor command is executed, Auto-QA will start behaving strangely (it will set the number of passes to 0, and start executing forever).

c. The cleanup code in the diagnostic was executed at the end of the first QA check, the Normal Start check. After executing the cleanup code, QA caused the diagnostic to continue running by starting the Multiple Pass check. That is, the /QA qualifier on the START or RUN commands cause the following to happen:

   i. The diagnostic is started and executes one pass. This means it executes the initialization code (doing the pass-zero things, such as allocating buffers and assigning channels), then executes all the tests in the appropriate section (the DEFAULT section by default).

   ii. Then, after executing the last test in that section, the Supervisor calls the initialization code again. This time, the pass-zero things will not be done and the ENDPASS routine will be called.

iii. The ENDPASS routine calls the diagnostic's cleanup code (where the buffers are usually deallocated and channels deassigned).

iv. Normally at this point (i.e., when NOT under control of QA), the Supervisor would return to command mode. However, QA will cause the ENDPASS routine to set the number of passes to QAMULTIPLEPASS (settable via a "SET QAMULTIPLEPASS" DS command) and start the Multiple Pass QA check.

v. This causes test one to execute again. Note that the initialization code is NOT executed in between the Normal Start check and Multiple Pass check. Thus, the buffers that were deallocated at the end of the Normal Start check (in the cleanup code) are no longer available to the diagnostic in the Multiple Pass check. This was causing several diagnostics (including the Cluster Exerciser) to fail.

d. The branch macros (i.e., $DS_BERROR, $DS_BLBS, $DS_BBS, $DS_BNFQ, etc.) will destroy the contents of R0. This happens because the branch macros expand into a Supervisor service routine, which is executed with a CALLS instruction. Since R0 is never saved across CALL instructions, the contents can not be guaranteed upon return from this routine.

e. The register values that are printed out when a QA error is detected are incomplete. That is, at most three of the registers are correct, with the others having a default value of zero. This led to some confusion.

f. The /TEST qualifier was ignored on the START or RUN commands when used in conjunction with the /QA qualifier.

g. The /SECTION qualifier did not work properly when used in conjunction with the /QA qualifier. The DEFAULT section is the only section that can be run when using the /QA qualifier.

4. Fixes to the Above Problems

The following describes the status of the above problems:

a. The HALT flag is now cleared when QA is running. As with the other Supervisor flags (e.g., IE1, IE2, IE3, TRACE, BELL, etc.), it is restored to its previous state (i.e., to the same setting it had before the START or RUN command was executed) after QA finishes.

b. A Control-C will now cause the diagnostic currently being QA'ed to be aborted only if Control-C has NOT been disabled by the diagnostic. That is, if the diagnostic has disabled C's, C's will be ignored when running under QA. If the diagnostic has not disabled C's, (they are enabled by default), the Supervisor will gain control

(before the diagnostic's Control-C handler) and will abort the diagnostic. Note that a CONTINUE Supervisor command in this context will not work.

c. The way the QA check routines execute the diagnostic has been fundamentally changed. Whereas before the sequence of checks could be thought of as executing a variable number of passes of the diagnostic, the sequence is now like STARTing the diagnostic a fixed number of times (the number of times depends on the number of QA checks).

For example, ASSUMING (note that even though there is none, this has been suggested as an enhancement) there was a new qualifier on the START (and RUN) command that would allow one to select which particular QA check routine to execute (such as /QA=NORMAL_START to execute ONLY the Normal Start QA check on the diagnostic), then the sequence of checks can now be thought of as being:

```
DS> START/QA=NORMAL_START
DS> START/QA=MULTIPLE_PASS
DS> START/QA=LOOP_ON_TEST
DS> START/QA=LOOP_ON_SUBTEST
DS> START/QA=RUN_BACKWARDS
DS> START/QA=ERROR_PHASE_ONE
```

That is, the diagnostic program is reSTARTed at the beginning of each QA check and executes as though a normal START Supervisor command had been executed. This implies that the initialization code is always done at the beginning of each check routine, then all the tests execute (a varying number of times depending on which QA check is executing), then the ENDPASS routine is executed causing the cleanup code to execute, and then the next QA check routine performs the same way. This new implementation should alleviate the deallocation problems encountered by several people.

d. The implementation of the branch macros has changed somewhat. Some diagnostic engineers felt that the $DS_BERROR and $DS_BNERROR macros should NOT alter the contents of R0 both when QA is not running, and when QA is running but not forcing errors (as happens in the Error Phase One QA check). In this version of the Supervisor and of the DIAG library, the $DS_BERROR and $DS_BNERROR macros will alter the contents of R0 only when QA is running and forcing errors.

For branch macros other than $DS_BERROR and $DS_BNERROR, the contents of R0 may change across the branch macro call. The main reason an exception was made in the case of the above two macros is that they existed prior to the QA enhancement and did not alter the contents of R0. Therefore, to be compatible with old diagnostics, it was felt that this should not change. However, with the new branch macros, R0 will change only in the low bit position (i.e., it may change from a one to a zero, or vice-versa). Note that this bit may change even when the /QA qualifier is not being used (i.e., in normal running of the diagnostic).

e. The register values printed out when a QA error is detected are now correct. The values printed are the values at the time of the call to the Supervisor Service routine that produced the QA error (e.g., a $DS_ERRHARD call).

f. The /TEST qualifier is no longer ignored when used in conjunction with the /QA qualifier. This allows one to run QA on a particular test in a diagnostic, or on a range of tests. However, if the /TEST qualifier is used, the Overall Error Summary table is not printed (see later section).

g. The /SECTION qualifier is still essentially ignored. This will be fixed in the next version of the Supervisor.

D. DIAGBOOT (DIAGNOSTIC SECONDARY BOOTSTRAP PROGRAM)

(DIAGBOOT has not changed from release 6.6 to 6.7 of VDS; this section is included for the convenience of those who have not seen the 6.6 release notes).

DIAGBOOT is loaded by VMB when a BOOT command is issued with register 5 bit 4 set. DIAGBOOT will sense the type of processor (11/780, 11/750, or 11/730) and load the appropriate VDS image (ESSAA, ECSAA, or ENSAA respectively). This new version of DIAGBOOT (version 4) supports the new VMS rooted systems (the old [SYSMAINT] becomes [SYS0.SYSMAINT], or [SYS1.SYSMAINT], etc.). The high nibble (bits 28 to 31) of R5 at boot time represent the hex digit which should be converted to ASCII and appended to the root directory name 'SYS'; this allows system root directories of 'SYS0' to 'SYSF' to co-exist on the same disk pack.

This new version of DIAGBOOT also allows the VDS image being booted to be non-contiguous. With previous versions of DIAGBOOT, VDS had to be contiguous.

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.8
6-Jul-1982

A. GENERAL ENCHANCEMENTS

1. ATTACH enforces device naming conventions

An additional Ptable descriptor statement ($DS_$NAME; see VAX-11 Diagnostic Design Guide, chapter 17 for information) allows VDS to verify the names given to devices when they are attached. For example, when an RK07 is ATTACHed, it must have a name of the generic format 'ggan', where 'gg' is the device generic prefix, 'DM', 'a' is the controller

letter of the RK611, and 'n' is the unit number.
Previously, VDS imposed no constraints on device names other than on whether a unit number was allowed, and on the magnitude of the unit number if allowed.

Because this is new functionality, and some old SCRIPTS or even diagnostics may not function under these new constraints (e.g., scripts which ATTACH an RK07 as 'DMC'), new commands have been implemented which will enable and disable this functionality under operator control. The commands are:

- o SET ENFORCE : This command enables enforcement of device name enforcement. It is the default condition when VDS starts, EXCEPT when running under APT.

- o CLEAR ENFORCE : This command disables enforcement of device name enforcment. It will cause VDS to behave as it did prior to this enhancement. CLEAR ENFORCE is the default condition when VDS is run under control of APT.

****NOTE****
These commands are intended only for EMERGENCY use. You should fix your script files and/or diagnostic programs immediately to conform to the device naming conventions.

2. AUTOSIZER (EVSBA) changes

   The autosizer (EVSBA V1.4) has been modified to generate the correct names for V6.8 of the Diagnostic Supervisor. The entire attach command line generated for each device is described in a separated document.

3. $DS_ERRPREP_x [num], [unit], [msgadr], [prlink], [p1--6]

   A new error message facility has been added to the Supervisor

for detecting device preparation errors. You should use this macro after detecting a device preparation error (e.g. disk not spinning) instead of another error message facility. The parameter list is the same as for the other error macros and is fully described in section 8.7 of the Vax-11 Diagnostic Design Guide (EK-1VAXD-TM-002).

B. GENERAL BUG FIXES

1. KERNEL STACK NOT VALID ABORTS

Previously, VDS allocated only 3 pages for KERNEL mode stack. In certain commands, notably a directory of an ODS stuctured disk (standalone only, with MM ON), these 3 pages could be overflowed, resulting in a KERNEL STACK NOT VALID ABORT, due to the page below the kernel stack being marked 'no access' to catch stack overflows. To minimize the chances of this sort of behavior, the kernel stack has been expanded to 10 pages for V6.8. This problem can not occur in on-line mode (under VMS).

2. TM03/TE16/TU45/TU77

The problem with not being able to access magtapes on this controller through the Supervisor has been solved in this released.

3. MACHINE CHECK LOGOUT

The machine check logout for the 11/780 (ESSAA) has been changed to show the error summary register in addition to the other parameters.

C. NOTES

1. TM78 SUPPORT

VDS theoretically supports MAGTAPE devices as file load devices. Currently, the only Magtape devices supported are TM03 and TS11 controllers. This release includes the beginnings of TM78 (TU78 drive) support: however, due to a lack of standalone hardware, it is not debugged or supported. This note is just to warn those who may try a file command (DIRECTORY, HELP, LOAD, RUN) from a TM78 magtape, or who look at the VDS map listing and see 'IFBTDRIVR' (which is the VDS TM78 readonly driver) that the TM78 code is undebugged and probably will not work as-is. However, TM78 support will be completed as soon as possible, and documented in a future release.

2. INTERRUPT STACK

The INTERRUPT stack size has be increased 4 pages so the priviledge exerciser can print out errors without failing.

3. CUSTOMER RUNNABLE DIAGNOSTICS (CRD)

The Supervisor has been enhanced to support CRD AUTO TEST on the 11/730 machine. For more information on CRD, see the "Nebula Customer Runnable Diagnostics Functional Specification" DTM-82-001-01, document.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.9 8-Nov-1982

1.0 BUG FIXES

- The following sequence of commands caused a "?? Translation not valid Fault ..." :

```
DS> SET MM ON
DS> ATTACH ...
```

  This occurred only in standalone mode, and only if a program had not been previously loaded with a supervisor LOAD command.

- A "^Z", typed at the console terminal in stand-alone mode, echoed garbage as a non-printing character. It has been fixed so that it now imitates the supervisor EXIT command, and returns control to the console program.

- The following error message appears when any file services (such as LOAD, RUN, DIRECTORY, HELP, and script files) are attempted with an RL11/RL02:

  %RMS-F-DEV, bad device, or inappropriate device type (R0=00018404)

  This is a bug only in version 6.8 in standalone mode, and is fixed in version 6.9.

- Also, only in version 6.8 and only for the 11/750, there is a bug which prevents the supervisor from loading large files from the console TU58 tape cartridge. Some small files could be loaded. Typically, the tape cartridge indicator light would remain lit for an excessively long time, and eventually the supervisor would respond with an error message. This is fixed in version 6.9.

- New versions of VMS which modify the system service transfer vector table experienced a problem running the supervisor in on-line mode. This occurred because the supervisor copied three pages of system service vectors for its own use, but the transfer vectors had been modified to branch into a fourth page. In order to accomodate other possible modifications to the VMS system service transfer vectors, the supervisor's transfer table was modified to jump directly to the actual VMS system service transfer vector table.

2.0 ENHANCEMENTS

- Support for the TM78/TU78 magnetic tape device as a file load device is now functional. This means that commands such as DIRECTORY, HELP, LOAD, and RUN, will now work when the TU78 is ATTACHed and a SET LOAD command has been executed.

- The SHOW MEMORY/BUFFER command has been enhanced to show the amount of buffer space in use by the diagnostic. The command displays the amounts in use in P0, P1, and S0 space (the number of pages in use). Previously, only the available buffer space was displayed. The SHOW MEMORY/ALL also displays this information.

- CUSTOMER RUNNABLE DIAGNOSTICS (CRD): VAX-11/730 MENU TEST

  The VAX Diagnostic Supervisor now supports VAX-11/730 MENU TEST (CRD MENU) as part of the CRD Package. CRD MENU is an off-line Menu-driven test feature of CRD. The purpose of CRD MENU is to provide the customer, or inexperienced operator, stand-alone VAX Diagnostic test capability for the VAX-11/730 'base' system package and, in particular, for VAX-11/730 Add-on options. CRD MENU, therefore, represents a complementary test package to VAX-11/730 AUTO TEST (see DTM-82-001-01 Nebula CRD Auto Test Functional Specification).

  CRD MENU, when invoked, will automatically determine CRD MENU supported hardware on the system and satisfy the necessary software requirements required for executing VAX Diagnostics. Through the Menu interface, the operator may select and test one supported hardware unit for test, or with the 'Test All' mode feature, select and test all supported system hardware. CRD MENU, therefore, provides full functional verification and fault isolation to the subsystem level of the system.

  CRD MENU is invoked by VAX-11/730 Console Command (typing 'T/M' to console prompt), at the successful completion of VAX-11/730 AUTO TEST (invoked by typing 'T' to console prompt), or from VAX Diagnostic Supervisor Command Level (typing 'CRD' to VDS CLI command prompt). However, the 'CRD' Supervisor command is not supported for use in Supervisor command files.

  Four new Supervisor commands have been added for CRD MENU Test. The commands are as follows:

  ```
  SET CRD/TRACE
  CLEAR CRD/TRACE
  SHOW CRD
  CRD
  ```

  The functions of these commands are explained in the ENSAB.DOC file.

For additional information:

- see ENSAB.DOC
- see the Functional Specification for VAX-11/730 MENU TEST
- Type "HELP ENSAB MENU" to VDS CLI Command prompt (this requires that the ENSAB.HLP file be on the load device).

3.0 MISCELLANEOUS

- A bug in the $D1_ERR_L MACRO in DIAG, which caused an invalid conditional directive, has been fixed.

4.0 KNOWN BUGS AND DEFICIENCIES

- The following commands will cause an error message if a comment follows the command:

```
SET ENFORCE
CLEAR ENFORCE
SET CRD/TRACE
CLEAR CRD/TRACE
SHOW CRD
CRD
```

For example,

```
DS> SET ENFORCE             ! Comment
?? Illegal command
   \SET ENFORCE             ! C\
```

All of the above commands except the 'CRD' command will still perform their correct function even though the Supervisor responds with "?? Illegal command". NOTE: the 'CRD' command will FAIL if the command is followed by a comment on the same line.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.10 3-Jan-1982

1.0 BUG FIXES

- The 6.9 VDS bug concerning comments on the following Supervisor commands has been fixed. All of the following commands now accept comments after the command:
  - SET ENFORCE
  - CLEAR ENFORCE
  - SET CRD/TRACE
  - CLEAR CRD/TRACE
  - SHOW CRD
  - CRD

2.0 ENHANCEMENTS

- Support for the TM80 magnetic tape device as a file load device is now functional. This means that commands such as DIRECTORY, HELP, LOAD, and RUN, will now work when the TU80 is ATTACHed and a SET LOAD command has been executed.
- A new supervisor service call, which loads the COMET PCS microcode and patch bits, has been added. The service is called by a supervisor macro, $DS_LOADPCS_x ($DS_LOADPCS_DEF, $DS_LOADPCS_G, $DS_LOADPCS_L, and $DS_LOADPCS_S for MACRO and $DS_LOADPCS for BLISS), with no arguments. The service returns DS$_NORMAL if the service is completed successfully, and DS$_NOPCS if PCS is not available on this CPU. Currently, the only diagnostic program which uses this call is ECKAX. This release does not contain PCS microcode which affects any instructions. It merely restores the PCS to a known state. If the CPU attempts to execute any of the PCS microcode, an error message will be printed out.

3.0 MISCELLANEOUS

-

4.0 KNOWN BUGS AND DEFICIENCIES

o

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.11
23-May-1983

1.0 BUG FIXES

o Implementation of Control-Z recognition in standalone mode was done incorrectly and has been removed, although no symptoms have been discovered.

2.0 ENHANCEMENTS

1. VDS 6.11 suoports booting and file loading from HSC50 on CI780.

3.0 MISCELLANEOUS

2. The device name for booting from the RP07 disk will now be DRan and not DBan

3. The driver for the TM78/TU78 was changed to have a longer delay between accesses of the ATTENTION register during the wait for command completion to prevent spurious operations caused by false signals. The driver was enhanced to work with tapes written at 6250 b.p.i and be usable on drives other that number 0.

o

4.0 KNOWN BUGS AND DEFICIENCIES

o

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.12
21-Jul-1983

1.0 NEW FEATURES

- The SET WIDTH command has been changed to work in online mode. The default is the same as for standalone mode, 80 characters.

- The SET PAGE command has been added to freeze terminal output after a specified number of lines. The user is then prompted for more output. The default value is zero, which indicates that no paging will be done. In order to determine a maximum value for a video terminal such that no terminal output will be lost, subtract 2 from the number of lines capable of being displayed at one time by the terminal. Two is the number of lines taken by the prompt message. Thus, for a VT100 with a 24 line screen, SET PAGE 22, will print the maximum amount of lines without any being lost.

- A change has been made so that a person creating a script can force the answer to a prompting question to be fetched from the user terminal instead of from the script. If, in the script, a prompting message is followed by "" instead of the answer to the prompt, then the vds will print that prompt on the terminal and wait for a response to be typed. then the vds will continue executing the script. full documentation on this feature will be available in the next version of the vax-11 diagnostic design guide.

- the rc11 p-table descriptor has been added, so that aztec is now supported with p-table descriptors for the rc11, rc25, and rcf25.

- a new vds service has been added, called $dsGetterm. this service, "get terminal characteristics", can be used in both standalone and user mode to obtain characteristics about the user terminal. you can determine the terminal type, whether it is a video or hardcopy terminal, other information. full documentation on this feature will be available in the next version of the vax-11 diagnostic design guide.

2.0 BUG FIXES

- o A problem with ECKAX hanging has been fixed. The CLRVEC routine was changed to reset the clock vector before resetting the clock. This compensates for the TOK hardware bug on 11/750s, which caused interrupts once in a blue moon when resetting the clock.

- o $DSPRINT macros now return an error status. SS$NORMAL, SS$BUFFEROVF, or SS$BADPARAM will be returned.

- o $DSINLOOP will return DS$NORMAL or DS$ERROR, instead of just setting or clear the low bit of R0.

3.0 MISCELLANEOUS

- o LCN Low Cost Nebula will be supported with ENSAA diagnostic Supervisor.

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.13
21-Nov-1983

1.0 NEW FEATURES

o

This is the first Diagnostic Supervisor to be linked under version 3 of VMS. Therefore it will not function correctly under those systems running version 2 or earlier versions of VMS.

o

The method of attaching the load path to the CI network has been modified to allow access to a user specified disk on the HSC50 controller. An example of the attach command lines for the 11/780 follows:

```
DS> ATTACH CI780 HUB PAA0 14 5 4
DS> ATTACH CI_NODE PAA0 CIA10 HSC50 10
DS> DISK CIA10 HSC50$DUA1
```

Please note the addition an extra attach command line for DISK

and the new form of the generic name. The name must be of the form 'name$DUannn' where 'name' is any alphanumeric string 1 to 6 characters long, and the '$DU' is required. The controller, 'a', is usually 'A' and 'nnn' is the disk unit number.

- A "SET MEMORY n" command has been added which allows the operator to change the amount of total physical memory available to the supervisor and diagnostic programs. This feature is useful when it is desired to verify that a diagnostic program and the supervisor will run in a smaller amount of memory than is actual y available on the system.

  The unit of measurement for "n" is pages. The value of "n" must be greater than or equal to zero, but less than the last value specified for total available physical memory. Total available physical memory may be reset to the actual available physical memory by using a value of zero for "n". A value greater than the actual available physical memory will be accepted but not used.

  The command will not be accepted in user mode. Also note that any devices attached or selected will be deselected and deattached after performing the command.

- A "INITPCS" command has been added. When this command is executed, the supervisor searches for the PCS750.BIN file on the console TU58. If it is found, the supervisor onboard image of the PCS750.BIN is

overwritten with the image found on the TU58. Otherwise, the supervisor onboard image remains the same. Then, the PCS is reloaded with the current PCS image.

This command will be useful when the supervisor being run does not contain the latest PCS750.BIN file, but the file is available on the console TU58. The command is not accepted when running the supervisor under VMS, and will only be accepted when running on an 11/750.

Also, this supervisor contains version CMT098 of PCS microcode patches.

2.0 BUG FIXES

- A bug in the $DS_EXIT_TEST macro, which caused an "[aborted]" message to be printed if TRACE was set and the low bit in R0 was clear, has been fixed. Now, the "[aborted]" message will only appear if the program does a $DS_ABORT_TEST

3.0 MISCELLANEOUS

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 6.14
20-Feb-1984

1 NEW FEATURES

- New device support is provided for DMZ32.

2 BUG FIXES

- Several supervisor BOOTDRIVRs use the TODR to perform a timeout function. For the 11/750, the TODR remains at zero if the battery backup has failed, and the system has been powered off. Thus, the TODR in this case was not able to

perform the timeout function correctly. However, the BOOTDRIVRs were written so that the code would always appear to succeed. This has been fixed by writing a 1 into the TODR (which starts the clock ticking) if the battery backup has failed and the system was powered down. This may fix some of the 'unkown error' problems with supervisor file services for 11/750s.

- The problem of terminal output from a user's control-C handler being suppressed has been fixed. This bug occurred only in standalone mode.

- When a diagnostic is aborted, the supervisor attempts to print out a user PC. When run under VMS, sometimes a PC in VMS system space was printed. This has been fixed.

3 MISCELLANEOUS

- The SET MEMORY command has been enhanced so that ptables for the console load device and the boot path devices are saved. Previously, all ptables were lost.

- The format of supervisor messages which inform the user that an interrupt, trap, or exception has occurred, has been changed to state that the interrupt, exception or trap occurred through an SCB vector rather than just a vector. For example,

?? Reserved/privileged instruction Fault through SCB
vector: 10(X)

instead of,

?? Reserved/privileged instruction Fault through
vector: 10(X)

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 7.0 18-Jul-1984

1 NEW FEATURES

o VMS V4 support

1. Long file name support.

Long filename support has been added to this version of VDS. A full file specification will have the capability for the filename and filetype each to have up to 39 characters maximum. The version number can have up to 5 characters. The directory names can have up to 39 characters and up to 7 nested sub-directories, each with up to 39 characters. The device names can have up to 15 characters and overall the full file specification is not to exceed 252 characters including punctuation.

EXAMPLE: DEVICE:[DIRECTORY]FILENAME.TYPE;VERSION
15 39.39.etc 39 39 5

2. Long device names.

Support code has been added to allow for device names up to 15 characters. VDS will scan 'device name' line for the long device name format. If found VDS will dynamically add device name buffer space to the Ptable,otherwise device names of the format 'ggan' will be handled as before. In both cases, ,the quadword descriptor ,HP$Q_DEVICE, will contain the length and string address of the device name and HP$T_DEVICE will contain the device name with the format of 'ggan'. The new device name format will be as follows: 'name$ggan',the total combination of which must not exceed 15 characters.

3. DIRECTORY/WIDE

In order to display long filenames, the DIRECTORY

command now has a /WIDE switch, which causes one filename per line to be printed. A DIRECTORY command without the /WIDE switch will display filenames in four 20-character columns, but for filenames longer than 20 characters, truncation will occur.

4. The value of the symbols IPL$_TIMER and $IPL_SYNCH is changed from 7 to 8 in VMS V4 library LIB. The supervisor, however, will continue to use the old value (7), because of possible conflicts between QIO services

and timer services. In order that diagnostics are encouraged to use the old value, the $IPLDEF macro has been included in the DIAG libraries.

5. This supervisor is not supported on versions of VMS prior to version 4.

2 BUG FIXES

- The $DS_CI_NODE ptable has been changed so that a response of KL10 to the Node_type question produces a value of 6 and a response of CINT a value of 7.

- Some internal supervisor errors of the form:

    ?? Software detected error in ...

  were causing the supervisor to hang in the middle of printing out the rest of this message. This has been fixed.

3 MISCELLANEOUS

- The supervisor now displays a legal statement at boot time regarding the right to use of the diagnostic supervisor and diagnostic programs.

    Two new VDS services, $DS_MOUNT and $DS_DISMOUNT, have been added.

- As of VMS Version 4, you cannot issue a VMS MOUNT command before running the VDS. If you do, and if the device's p-table indicates that the device should be allocated, then the VDS SELECT command will fail. This failure occurs because the SELECT command will try to allocate the device, but you cannot allocate a device that has already been mounted (in VMS V3 you could, if the device had been mounted by you and not someone else).

    Any diagnostic programs that require a device to be allocated AND mounted before it is tested will not work with this release. We know of only one such diagnostic, EVRMD. We are in the process of correcting the problem. The correction will be in a future release.

- When the supervisor is run in online mode (under VMS), the default load path has been changed to be equivalent to SYS$SYSROOT:[SYSMAINT]. Previously, a bug in the supervisor caused the default load device to be the translated value of SYS$DISK from the process name table rather than from the system name table. Normally, SYS$SYSROOT:[SYSMAINT] is the equivalent of SYS$MAINTENANCE, and the supervisor will be changed in some future release to fully translate SYS$MAINTENANCE.

Object Module Synopsis

```
%LINK, 2 undefined symbols:
%LINK,  DS$GB_CI_INIT (Weak Reference)
%LINK,  FIL$GQ_CACHE (Weak Reference)
%LINK,  FIL$GT_TOPSYS (Weak Reference)
%LINK,  SYS$DISMOU
%LINK,  SYS$MOUNT
%LINK,  XDELBP1 (Weak Reference)
%LINK,  XDELTBIT (Weak Reference)
```

Module Name	Ident	Bytes	File	Creation Date	Creator
KERNEL	07-97	6027	DRB1:[DS.WORK]DS.OLB;18	23-JUL-1984 15:46	VAX-11 Macro V03-01
ACPFDT	06-03	251	DRB1:[DS.WORK]DS.OLB;18	3-FEB-1984 09:00	VAX-11 Macro V03-01
ANSI	07-07	2482	DRB1:[DS.WORK]DS.OLB;18	6-Jun-1984 17:53	VAX-11 Bliss-32 V4.0-742
APT	07-10	501	DRB1:[DS.WORK]DS.OLB;18	8-MAR-1984 12:51	VAX-11 Macro V03-01
ASSIGN	05-02	303	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:04	VAX-11 Macro V03-01
ASTCON	V02-001	44	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:07	VAX-11 Macro V03-01
ASTDEL	07-07	433	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 15:17	VAX-11 Macro V03-01
ATTACH	07-27	5092	DRB1:[DS.WORK]DS.OLB;18	7-JUN-1984 08:40	VAX-11 Macro V03-01
BOOTDRIVR	V02-017	527	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:14	VAX-11 Macro V03-01
BOOTIO	06-02	84	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:15	VAX-11 Macro V03-01
BUFCTL	05-03	100	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:16	VAX-11 Macro V03-01
BUFFER	07-06	223	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:19	VAX-11 Macro V03-01
CALLFRAME	01-01	464	DRB1:[DS.WORK]DS.OLB;18	9-Jan-1984 14:07	VAX-11 Bliss-32 V4.0-742
CANCEL	05-04	265	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:21	VAX-11 Macro V03-01
CHAN730	06-11	1330	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 11:29	VAX-11 Macro V03-01
CHMK	07-07	146	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 14:18	VAX-11 Macro V03-01
CLI	07-80	7752	DRB1:[DS.WORK]DS.OLB;18	7-JUN-1984 08:41	VAX-11 Macro V03-01
CLOCK	07-30	1172	DRB1:[DS.WORK]DS.OLB;18	17-FEB-1984 13:49	VAX-11 Macro V03-01
COMDRVSUB	06.01	245	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 14:39	VAX-11 Macro V03-01
CONFIG	06-19	2091	DRB1:[DS.WORK]DS.OLB;18	10-JUL-1984 16:21	VAX-11 Macro V03-01
CONSOLE	07-23	1709	DRB1:[DS.WORK]DS.OLB;18	23-JUL-1984 16:06	VAX-11 Macro V03-01
CRDIMAGE	02-11	5699	DRB1:[DS.WORK]DS.OLB;18	9-Jan-1984 14:09	VAX-11 Bliss-32 V4.0-742
CSDDBTDRV	06-11	649	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:10	VAX-11 Macro V03-01
DSVECS	01-09	112	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 16:24	VAX-11 Macro V03-01
CVRTIM	V02-02	1036	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 15:10	VAX-11 Macro V03-01
CVTFILNAM	06-02	214	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 15:11	VAX-11 Macro V03-01
DASSGN	05-02	189	DRB1:[DS.WORK]DS.OLB;18	3-FEB-1984 09:05	VAX-11 Macro V03-01
DEBUG	07-15	4004	DRB1:[DS.WORK]DS.OLB;18	23-JUL-1984 15:36	VAX-11 Macro V03-01
DEVALC	05-03	405	DRB1:[DS.WORK]DS.OLB;18	20-JUL-1984 16:15	VAX-11 Macro V03-01

```
%LINK, undefined symbol SYS$MOUNT referenced
        in psect SEP offset %X00000172
        in module DEVALC file DRB1:[DS.WORK]DS.OLB;18
%LINK, undefined symbol SYS$DISMOU referenced
        in psect SEP offset %X0000018F
        in module DEVALC file DRB1:[DS.WORK]DS.OLB;18
```

DEVICE	06-01	175	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 15:19	VAX-11 Macro V03-01
DIRECTORY	07-17	2870	DRB1:[DS.WORK]DS.OLB;18	6-Jun-1984 17:54	VAX-11 Bliss-32 V4.0-742
DISPAT	07-37	2089	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:14	VAX-11 Macro V03-01
DLBTDRIVR	V02-002	312	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 15:28	VAX-11 Macro V03-01
DMBTDRIVR	V02-002	315	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 15:30	VAX-11 Macro V03-01
DRINT	V01	29	DRB1:[DS.WORK]DS.OLB;18	3-JAN-1984 16:05	VAX-11 Macro V03-01

Module Name	Ident	Bytes	File	Creation Date	Creator
DSLOAD	07-09	604	DRB1:[DS.WORK]DS.OLB;18	23-Jul-1984 15:52	VAX-11 Bliss-32 V4.0-742
DQBTDRIVR	V02-001	333	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:15	VAX-11 Macro V03-01
ENTRY	07-35	2985	DRB1:[DS.WORK]DS.OLB;18	13-JUL-1984 14:06	VAX-11 Macro V03-01
ERROR	07-24	3425	DRB1:[DS.WORK]DS.OLB;18	24-APR-1984 09:24	VAX-11 Macro V03-01
EVENT	06-01	241	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 08:57	VAX-11 Bliss-32 V4.0-742
EXCEPT	07-24	3701	DRB1:[DS.WORK]DS.OLB;18	26-JUN-1984 09:39	VAX-11 Macro V03-01
FAO	03-02	865	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 09:01	VAX-11 Macro V03-01
FLAGS	07-13	582	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 09:08	VAX-11 Macro V03-01
FILEREAD	06-02	2129	DRB1:[DS.WORK]DS.OLB;18	1-JUN-1984 09:38	VAX-11 Macro V03-01
FRKCTL	06-03	102	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 09:09	VAX-11 Macro V03-01
GETTIM	01	26	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 09:10	VAX-11 Macro V03-01
GTCHAN	02-01	136	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 09:11	VAX-11 Macro V03-01
HELP	06-10	2528	DRB1:[DS.WORK]DS.OLB;18	6-Jun-1984 17:54	VAX-11 Bliss-32 V4.0-742
IOBASE	12-46	9799	DRB1:[DS.WORK]DS.OLB;18	23-JUL-1984 15:40	VAX-11 Macro V03-01
IOPOST	05-04	979	DRB1:[DS.WORK]DS.OLB;18	3-FEB-1984 09:08	VAX-11 Macro V03-01
IOSNPG	06-03	767	DRB1:[DS.WORK]DS.OLB;18	3-FEB-1984 09:12	VAX-11 Macro V03-01
IOSPGD	05-02	402	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 12:01	VAX-11 Macro V03-01
IOSRAM	05-02	373	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 12:02	VAX-11 Macro V03-01
LOAD	07-21	875	DRB1:[DS.WORK]DS.OLB;18	20-JUL-1984 14:16	VAX-11 Macro V03-01
LOOMAP	05-02	255	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 13:39	VAX-11 Macro V03-01
LOOP	07-19	815	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 13:33	VAX-11 Macro V03-01
MCHK730	06-07	1163	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 13:43	VAX-11 Bliss-32 V4.0-742
MBAINT	01	180	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 13:35	VAX-11 Macro V03-01
MEMALC	07-06	707	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:31	VAX-11 Macro V03-01
MEMMGT	06-29	2140	DRB1:[DS.WORK]DS.OLB;18	22-JUN-1984 10:04	VAX-11 Macro V03-01
ODS	01-04	972	DRB1:[DS.WORK]DS.OLB;18	24-May-1984 16:16	VAX-11 Bliss-32 V4.0-742
ONLY730	07-12	663	DRB1:[DS.WORK]DS.OLB;18	23-JUL-1984 16:09	VAX-11 Macro V03-01
PARAM	6.5-11	1575	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 15:30	VAX-11 Macro V03-01
PARSE	07-14	1155	DRB1:[DS.WORK]DS.OLB;18	7-JUN-1984 08:47	VAX-11 Macro V03-01
PCSLOAD	06-03	77	DRB1:[DS.WORK]DS.OLB;18	9-Apr-1984 15:15	VAX-11 Bliss-32 V4.0-742
PRINT	07-19	1748	DRB1:[DS.WORK]DS.OLB;18	23-JUL-1984 15:50	VAX-11 Macro V03-01
PROBE	01-05	184	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 15:43	VAX-11 Bliss-32 V4.0-742
PUBTDRIVR	06-01	768	DRB1:[DS.WORK]DS.OLB;18	24-JAN-1984 10:08	VAX-11 Macro V03-01
QACHECKS	6.5-03	2098	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 15:59	VAX-11 Bliss-32 V4.0-742
QAFLAGS	6.6-04	484	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 16:02	VAX-11 Bliss-32 V4.0-742
QAMAIN	1-08	2799	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 16:03	VAX-11 Bliss-32 V4.0-742
QIO	07-18	2875	DRB1:[DS.WORK]DS.OLB;18	26-APR-1984 10:41	VAX-11 Macro V03-01
QIOFDT	05-03	479	DRB1:[DS.WORK]DS.OLB;18	3-FEB-1984 09:14	VAX-11 Macro V03-01
QIOREQ	01-02	564	DRB1:[DS.WORK]DS.OLB;18	3-FEB-1984 09:16	VAX-11 Macro V03-01
RELOCDRV	01	164	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 16:11	VAX-11 Macro V03-01
RMS	06.23	2960	DRB1:[DS.WORK]DS.OLB;18	13-Apr-1984 10:32	VAX-11 Bliss-32 V4.0-742
RT11	01-04	835	DRB1:[DS.WORK]DS.OLB;18	4-Jan-1984 16:15	VAX-11 Bliss-32 V4.0-742
SCB	07-29	1093	DRB1:[DS.WORK]DS.OLB;18	4-JUN-1984 13:37	VAX-11 Macro V03-01
SCRIPT	07-18	1708	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:42	VAX-11 Macro V03-01
SHOWMEMORY	06-06	1961	DRB1:[DS.WORK]DS.OLB;18	17-Feb-1984 14:29	VAX-11 Bliss-32 V4.0-742
STAR1	07-41	1868	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:43	VAX-11 Macro V03-01
TSBTDRIVR	V01-02	522	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:47	VAX-11 Macro V03-01
MUBTDRIVR	06-9	990	DRB1:[DS.WORK]DS.OLB;18	9-APR-1984 16:33	VAX-11 Macro V03-01
UNWIND	01-1	297	DRB1:[DS.WORK]DS.OLB;18	4-JAN-1984 16:36	VAX-11 Macro V03-01
VER730	7.0	106	DRB1:[DS.WORK]DS.OLB;18	26-JUL-1984 09:25	VAX-11 Macro V03-01
VMSDUMMY	05-02	30	DRB1:[DS.WORK]DS.OLB;18	5-JAN-1984 09:19	VAX-11 Macro V03-01
SYS$IODEF	V03-001	0	SYS$SYSROOT:[SYSLIB]STARLET.OLB;2	3-JUN-1984 11:29	VAX-11 Macro V03-01

Module Name	Ident	Bytes	File	Creation Date	Creator
LIB$CVT_ATB	V03-001	122	SYS$SYSROOT:[SYSLIB]STARLET.OLB;2	3-JUN-1984 14:47	VAX-11 Macro V03-01
SYS$PRDEF	V03-000	0	SYS$SYSROOT:[SYSLIB]STARLET.OLB;2	3-JUN-1984 12:30	VAX-11 Macro V03-01
SYS$SSDEF	V03-000	0	SYS$SYSROOT:[SYSLIB]STARLET.OLB;2	3-JUN-1984 13:01	VAX-11 Macro V03-01
SYS$P1_VECTOR	V03-040	0	SYS$SYSROOT:[SYSLIB]STARLET.OLB;2	3-JUN-1984 12:04	VAX-11 Macro V03-01

Image Section Synopsis

Cluster	Type	Pages	Base Addr	Disk VBN	PFC	Protection and Paging	Global Sec. Name	Match	Majorid	Minorid
DEFAULT_CLUSTER	0	67	00010000	2	0	READ ONLY				
	0	14	00018600	69	0	READ WRITE COPY ON REF				
	0	122	0001A200	83	0	READ ONLY				
	0	24	00029600	205	0	READ WRITE COPY ON REF				
	0	1	0002C600	229	0	READ WRITE FIXUP VECTORS				
	253	20	7FFFD800	0	0	READ WRITE DEMAND ZERO				

Key for special characters above:

R	Relocatable
P	Protected

Program Section Synopsis

Psect Name	Module Name	Base	End	Length		Align	Attributes
$BASE		00010000	00010B27	00000B28 (	2856.)	QUAD 3	NOPIC,USR,CON,REL,LCL, SHR,NOEXE, RD,NOWRT,NOVEC
	ENTRY	00010000	00010B27	00000B28 (	2856.)	QUAD 3	
DATA		00010B40	000185C1	00007A82 (	31362.)	2 ** 5	NOPIC,USR,CON,REL,LCL, SHR,NOEXE, RD,NOWRT,NOVEC
	KERNEL	00010B40	00010CAF	00000170 (	368.)	BYTE 0	
	ANSI	00010CB0	00010CC7	00000018 (	24.)	LONG 2	
	APT	00010CC8	00010CEC	00000025 (	37.)	LONG 2	
	ASTDEL	00010CF0	00010D02	00000013 (	19.)	LONG 2	
	ATTACH	00010D04	00010FE1	000002DE (	734.)	LONG 2	
	CALLFRAME	00010FE4	000110C4	000000E1 (	225.)	LONG 2	
	CHAN730	000110C8	00011120	00000059 (	89.)	LONG 2	
	CHMK	00011124	00011128	00000005 (	5.)	LONG 2	
	CLI	00011129	000120F4	00000FCC (	4044.)	BYTE 0	
	CLOCK	000120F5	000120FA	00000006 (	6.)	BYTE 0	
	CONFIG	000120FC	0001219A	0000009F (	159.)	LONG 2	
	CONSOLE	0001219C	000122D8	0000013D (	317.)	LONG 2	
	CRDIMAGE	000122DC	00012FC3	00000CE8 (	3304.)	LONG 2	
	DEBUG	00012FC4	000132AB	000002E8 (	744.)	BYTE 0	
	DEVICE	000132AC	000132B2	00000007 (	7.)	BYTE 0	
	DIRECTORY	000132B4	0001347B	000001C8 (	456.)	LONG 2	
	DISPAT	0001347C	00013716	0000029B (	667.)	LONG 2	
	DSLOAD	00013718	00013777	00000060 (	96.)	LONG 2	
	ENTRY	00013778	000137BF	00000048 (	72.)	QUAD 3	
	ERROR	000137C0	00014057	00000898 (	2200.)	BYTE 0	
	EXCEPT	00014058	000147BE	00000767 (	1895.)	LONG 2	
	FAO	000147C0	000147FC	0000003D (	61.)	LONG 2	
	FLAGS	000147FD	000148D6	000000DA (	218.)	BYTE 0	
	HELP	000148D8	0001497E	000000A7 (	167.)	LONG 2	
	IOBASE	0001497F	00016B91	00002213 (	8723.)	BYTE 0	
	IOPOST	00016B94	00016BA4	00000011 (	17.)	LONG 2	
	LOAD	00016BA5	00016C08	00000064 (	100.)	BYTE 0	
	LOOP	00016C09	00016CAC	000000A4 (	164.)	BYTE 0	
	MCHK730	00016CB0	000170CF	00000420 (	1056.)	LONG 2	
	MEMALC	000170D0	000170EB	0000001C (	28.)	BYTE 0	
	MEMMGT	000170EC	00017129	0000003E (	62.)	LONG 2	
	ONLY730	0001712C	0001713B	00000010 (	16.)	LONG 2	
	PARAM	0001713C	00017278	0000013D (	317.)	BYTE 0	
	PCSLOAD	0001727C	000172AE	00000033 (	51.)	LONG 2	
	PRINT	000172AF	000172BE	00000010 (	16.)	BYTE 0	
	QACHECKS	000172C0	0001765A	0000039B (	923.)	LONG 2	
	QAFLAGS	0001765C	00017702	000000A7 (	167.)	LONG 2	
	QAMAIN	00017704	00017B8A	00000487 (	1159.)	LONG 2	
	QIO	00017B8C	00017C83	000000F8 (	248.)	LONG 2	
	RMS	00017C84	00017F33	000002B0 (	688.)	LONG 2	
	SCB	00017F34	00017F37	00000004 (	4.)	LONG 2	
	SCRIPT	00017F38	00017FA1	0000006A (	106.)	BYTE 0	
	SHOWMEMORY	00017FA4	00018318	00000375 (	885.)	LONG 2	
	START	00018319	00018536	0000021E (	542.)	BYTE 0	

Psect Name	Module Name	Base	End	Length		Align	Attributes
DATA		00010B40	000185C1	00007A82	(31362.)	2 ** 5	NOPIC,USR,CON,REL,LCL, SHR,NOEXE, RD,NOWRT,NOVEC
	TSBTDRIVR	00018540	0001855F	00000020	(32.)	2 ** 5	
	VER730	00018560	000185C1	00000062	(98.)	LONG 2	
WORK		00018600	0001A168	00001B69	(7017.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	KERNEL	00018600	000190E9	00000AEA	(2794.)	LONG 2	
	ATTACH	000190EC	000190F5	0000000A	(10.)	LONG 2	
	BUFFER	000190F8	00019117	00000020	(32.)	LONG 2	
	CHAN730	00019118	00019208	000000F1	(241.)	LONG 2	
	CHMK	0001920C	0001920F	00000004	(4.)	LONG 2	
	CLI	00019210	0001965C	0000044D	(1101.)	LONG 2	
	CLOCK	000196C0	000196C7	00000068	(104.)	LONG 2	
	CONSOLE	000196C8	000196CF	00000008	(8.)	LONG 2	
	CRDIMAGE	000196D0	00019753	00000084	(132.)	LONG 2	
	DSVECS	00019754	000197C3	00000070	(112.)	LONG 2	
	DEBUG	000197C4	0001994C	00000189	(393.)	LONG	
	DIRECTORY	00019950	00019959	0000000A	(10.)	LONG	
	DISPAT	0001995C	00019963	00000008	(8.)	LONG 2	
	ERROR	00019964	00019993	00000030	(48.)	LONG 2	
	EXCEPT	00019994	000199A7	00000014	(20.)	LONG 2	
	HELP	000199A8	000199D5	0000002E	(46.)	LONG 2	
	IOBASE	000199D8	00019DF3	0000041C	(1052.)	LONG 2	
	LOAD	00019DF4	00019F13	00000120	(288.)	LONG 2	
	MEMALC	00019F14	00019F2F	0000001C	(28.)	LONG 2	
	MEMMGT	00019F30	00019F4C	0000001D	(29.)	LONG 2	
	ONLY730	00019F50	00019F53	00000004	(4.)	LONG 2	
	PARSE	00019F54	00019F5B	00000008	(8.)	LONG 2	
	PRINT	00019F5C	00019FE1	00000086	(134.)	LONG 2	
	QACHECKS	00019FE4	00019FE5	00000002	(2.)	LONG 2	
	QAFLAGS	00019FE6	00019FED	00000006	(6.)	LONG 2	
	QAMAIN	00019FF0	0001A024	00000035	(53.)	LONG 2	
	QIO	0001A028	0001A034	0000000D	(13.)	LONG 2	
	RMS	0001A038	0001A047	00000010	(16.)	LONG 2	
	SCB	0001A048	0001A098	00000051	(81.)	LONG 2	
	SCRIPT	0001A09C	0001A164	000000C9	(201.)	LONG 2	
	SHOWMEMORY	0001A168	0001A168	00000001	(1.)	LONG 2	
CODE		0001A200	000293D2	0000F1D3	(61907.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	KERNEL	0001A200	0001AD30	00000B31	(2865.)	LONG 2	
	ANSI	0001AD34	0001B6CD	0000099A	(2458.)	LONG 2	
	APT	0001B6D0	0001B89F	000001D0	(464.)	LONG 2	
	ASTCON	0001B8A0	0001B8CB	0000002C	(44.)	LONG 2	
	ASTDEL	0001B8CC	0001BA69	0000019E	(414.)	LONG 2	
	ATTACH	0001BA6C	0001CB67	000010FC	(4348.)	LONG 2	
	BOOTIO	0001CB68	0001CBBB	00000054	(84.)	LONG 2	
	BUFFER	0001CBBC	0001CC7A	000000BF	(191.)	BYTE 0	
	CALLFRAME	0001CC7C	0001CD6A	000000EF	(239.)	LONG 2	
	CHAN730	0001CD6C	0001D153	000003E8	(1000.)	LONG 2	
	CHMK	0001D154	0001D1DC	00000089	(137.)	LONG 2	
	CLI	0001D1DD	0001DC0B	00000A2F	(2607.)	BYTE 0	
	CLOCK	0001DC0C	0001E031	00000426	(1062.)	LONG 2	
	CONFIG	0001E034	0001E7BF	0000078C	(1932.)	LONG 2	

Psect Name	Module Name	Base	End	Length		Align	Attributes
CODE		0001A200	000293D2	0000F1D3	(61907.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	CONSOLE	0001E7C0	0001ED27	00000568	(1384.)	LONG 2	
	CRDIMAGE	0001ED28	0001F5FE	000008D7	(2263.)	LONG 2	
	CVTFILNAM	0001F600	0001F6D5	000000D6	(214.)	LONG 2	
	DEBUG	0001F6D6	00020208	00000B33	(2867.)	BYTE 0	
	DEVICE	00020209	000202B0	000000A8	(168.)	BYTE 0	
	DIRECTORY	000202B4	00020C17	00000964	(2404.)	LONG 2	
	DISPAT	00020C18	0002119D	00000586	(1414.)	LONG 2	
	DSLOAD	000211A0	0002139B	000001FC	(508.)	LONG 2	
	ENTRY	0002139C	000213D4	00000039	(57.)	BYTE 0	
	ERROR	000213D5	0002186D	00000499	(1177.)	BYTE 0	
	EVENT	00021870	00021960	000000F1	(241.)	LONG 2	
	EXCEPT	00021964	0002205D	000006FA	(1786.)	LONG 2	
	FAO	00022060	00022383	00000324	(804.)	LONG 2	
	FLAGS	00022384	000224EF	0000016C	(364.)	BYTE 0	
	FILERFAD	000224F0	00022D40	00000851	(2129.)	LONG 2	
	HELP	00022D44	0002364E	0000090B	(2315.)	LONG 2	
	IOPOST	00023650	00023A11	000003C2	(962.)	LONG 2	
	LOAD	00023A12	00023BF8	000001E7	(487.)	BYTE 0	
	LOOP	00023BF9	00023E83	0000028B	(651.)	BYTE 0	
	MCHK730	00023E84	00023EEE	0000006B	(107.)	LONG 2	
	MEMALC	00023EEF	00024179	0000028B	(651.)	BYTE 0	
	MEMMGT	0002417C	0002497C	00000801	(2049.)	LONG 2	
	ODS	00024980	00024D4B	000003CC	(972.)	LONG 2	
	ONLY730	00024D4C	00024FCE	00000283	(643.)	LONG 2	
	PARAM	00024FD0	000254B9	000004EA	(1258.)	LONG 2	
	PARSE	000254BA	00025934	0000047B	(1147.)	BYTE 0	
	PCSLOAD	00025938	00025951	0000001A	(26.)	LONG 2	
	PRINT	00025952	00025F8F	0000063E	(1598.)	BYTE 0	
	QACHECKS	00025F90	00026424	00000495	(1173.)	LONG 2	
	QAFLAGS	00026428	0002655E	00000137	(311.)	LONG 2	
	QAMAIN	00026560	00026B92	00000633	(1587.)	LONG 2	
	QIO	00026B94	000275C9	00000A36	(2614.)	LONG 2	
	RMS	000275CC	00027E9B	000008D0	(2256.)	LONG 2	
	RT11	00027E9C	000281DE	00000343	(835.)	LONG 2	
	SCB	000281E0	000284CF	000002F0	(752.)	LONG 2	
	SCRIPT	000284D0	00028A48	00000579	(1401.)	BYTE 0	
	SHOWMEMORY	00028A4C	00028E7E	00000433	(1075.)	LONG 2	
	START	00028E7F	000293AC	0000052E	(1326.)	BYTE 0	
	VER730	000293AD	000293B4	00000008	(8.)	BYTE 0	
	VMSDUMMY	000293B5	000293D2	0000001E	(30.)	BYTE 0	
_LIB$CODE		000293D4	0002944D	0000007A	(122.)	LONG 2	PIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	LIB$CVT_ATB	000293D4	0002944D	0000007A	(122.)	LONG 2	
$SCB		00029600	000296FF	00000100	(256.)	PAGE 9	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD, WRT,NOVEC
	SCB	00029600	000296FF	000001C0	(256.)	PAGE 9	
BOOTDRIVR_1		00029700	000298A1	000001A2	(418.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	00029700	000298A1	000001A2	(418.)	BYTE 0	
BOOTDRIVR_2		00029A00	0002A9C5	00000FC6	(4038.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC

Psect Name	Module Name	Base	End	Length		Align	Attributes
BOOTDRIVR_2		00029A00	0002A9C5	00000FC6	(4038.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	CSDDBTDRV	00029A00	00029C70	00000271	(625.)	BYTE 0	
	DLBTDRIVR	00029C71	00029D90	00000120	(288.)	BYTE 0	
	DMBTDRIVR	00029D91	00029EB3	00000123	(291.)	BYTE 0	
	DQBTDRIVR	00029EB4	00029FE8	00000135	(309.)	BYTE 0	
	PUBTDRIVR	0002A000	0002A2E7	000002E8	(744.)	PAGE 9	
	TSBTDRIVR	0002A2E8	0002A4B9	000001D2	(466.)	BYTE 0	
	MUBTDRIVR	0002A600	0002A9C5	000003C6	(966.)	PAGE 9	
BOOTDRIVR_3		0002A9C6	0002A9C6	00000000	(0.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	0002A9C6	0002A9C6	00000000	(0.)	BYTE 0	
BOOTDRIVR_4		0002A9C6	0002AA6D	000000A8	(168.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	CSDDBTDRV	0002A9C6	0002A9DD	00000018	(24.)	BYTE 0	
	DLBTDRIVR	0002A9DE	0002A9F5	00000018	(24.)	BYTE 0	
	DMBTDRIVR	0002A9F6	0002AA0D	00000018	(24.)	BYTE 0	
	DQBTDRIVR	0002AA0E	0002AA25	00000018	(24.)	BYTE 0	
	PUBTDRIVR	0002AA26	0002AA3D	00000018	(24.)	BYTE 0	
	TSBTDRIVR	0002AA3E	0002AA55	00000018	(24.)	BYTE 0	
	MUBTDRIVR	0002AA56	0002AA6D	00000018	(24.)	BYTE 0	
BOOTDRIVR_5		0002AA6E	0002AA71	00000004	(4.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	0002AA6E	0002AA71	00000004	(4.)	BYTE 0	
BOOTDRIVR_6		0002AA72	0002AADA	00000069	(105.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	0002AA72	0002AADA	00000069	(105.)	BYTE 0	
SEP		0002AADC	0002C570	00001A95	(6805.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD, WRT,NOVEC
	ACPFDT	0002AADC	0002ABD6	000000FB	(251.)	LONG 2	
	ASSIGN	0002ABD8	0002AD06	0000012F	(303.)	LONG 2	
	BUFCTL	0002AD08	0002AD6B	00000064	(100.)	LONG 2	
	CANCEL	0002AD6C	0002AE74	00000109	(265.)	LONG 2	
	COMDRVSUB	0002AE78	0002AF6C	000000F5	(245.)	LONG 2	
	CVRTIM	0002AF70	0002B37B	0000040C	(1036.)	LONG 2	
	DASSGN	0002B37C	0002B438	000000BD	(189.)	LONG 2	
	DEVALC	0002B43C	0002B5D0	00000195	(405.)	LONG 2	
	DRINT	0002B5D4	0002B5F0	0000001D	(29.)	LONG 2	
	FRKCTL	0002B5F4	0002B659	00000066	(102.)	LONG 2	
	GETTIM	0002B65C	0002B675	0000001A	(26.)	LONG 2	
	GTCHAN	0002B678	0002B6FF	00000088	(136.)	LONG 2	
	IOBASE	0002B700	0002B717	00000018	(24.)	LONG 2	
	IOSNPG	0002B718	0002BA16	000002FF	(767.)	LONG 2	
	IOSPGD	0002BA18	0002BBA9	00000192	(402.)	LONG 2	
	IOSRAM	0002BBAC	0002BD20	00000175	(373.)	LONG 2	
	LODMAP	0002BD24	0002BE22	000000FF	(255.)	LONG 2	
	MBAINT	0002BE24	0002BED7	000000B4	(180.)	LONG 2	
	PROBE	0002BED8	0002BF8F	000000B8	(184.)	LONG 2	
	QIOFDT	0002BF90	0002C16E	000001DF	(479.)	LONG 2	
	QIOREQ	0002C170	0002C3A3	00000234	(564.)	LONG 2	
	RELOCDRV	0002C3A4	0002C447	000000A4	(164.)	LONG 2	
	UNWIND	0002C448	0002C570	00000129	(297.)	LONG 2	

Psect Name	Module Name	Base	End	Length	Align	Attributes
Y$EXEPAGED		0002C571	0002C571	00000000 (0.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ASTCON	0002C571	0002C571	00000000 (0.)	BYTE 0	
_LAST		0002C600	0002C600	00000000 (0.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	KERNEL	0002C600	0002C600	00000000 (0.)	PAGE 9	

Symbol Cross Reference

Symbol	Value	Defined By	Referenced By ...		
AAT_ARITH	000141C3-R	EXCEPT			
ACC$HANDLER	0002BF4B-R	PROBE			
ACP$ACCESS	0002AADC-R	ACPFDT	ENTRY		
ACP$ACCESSNET	0002AADC-R	ACPFDT			
ACP$DEACCESS	0002AAE7-R	ACPFDT	ENTRY		
ACP$MODIFY	0002AAF2-R	ACPFDT	ENTRY		
ACP$MOUNT	0002AAFD-R	ACPFDT	ENTRY		
ACP$READBLK	0002AB08-R	ACPFDT	ENTRY		
ACP$WRITEBLK	0002AB25-R	ACPFDT	ENTRY		
ACT_CONT	0001D4C6-R	CLI			
ACT_DEFAULT_DBG	0001D4D9-R	CLI	START		
ANSI$CACHEVBN	0001B30D-R	ANSI			
ANSI$CLOSE	0001B2DC-R	ANSI			
ANSI$GET	0001B4D3-R	ANSI			
ANSI$OPEN	0001AD34-R	ANSI	RMS		
ANSI$READ	0001B412-R	ANSI			
APSLMNE	000131BC-R	DEBUG	EXCEPT	MCHK730	
APT	0001A297-R	KERNEL	ENTRY		
APT_COM	0001B6D0-R	APT	CLI		
APT_DONE	0001B891-R	APT			
APT_MSG	0001B888-R	APT	CLI	ERROR	
AX_BUFF	0002C600-R	KERNEL	SHOWMEMORY		
BEGIN	0001A91E-R	KERNEL	CLI	DISPAT	DSVECS
			LOAD	LOOP	SCRIPT
			START		
BEGIN_BLISS	0001A91E-R	KERNEL	CRDIMAGE	DIRECTORY	SHOWMEMORY
BOO$AL_VECTOR	00029700-R	BOOTDRIVR	DIRECTORY	RMS	
BOO$DRIVER	000298A2-R	BOOTDRIVR			
BOO$IMAGE_ATT	0001CB88-R	BOOTIO			
BOO$MAP	000297F1-R	BOOTDRIVR			
BOO$QIO	00029716-R	BOOTDRIVR	ANSI	DIRECTORY	ODS
			RT11		
BOO$SELECT	0002AA72-R	BOOTDRIVR	DIRECTORY	RMS	
BOOT	0001A200-R	KERNEL	ENTRY		
BOOT$L_ADP	00018DD4-R	KERNEL	CONFIG		
BOOT$L_CSR	00018DD8-R	KERNEL	CONFIG		
BOOT$L_DEVTYP	00018DE8-R	KERNEL	CONFIG		
BOOT$L_R2	00018DE0-R	KERNEL	CONFIG		
BOOT$L_R5	00018DE4-R	KERNEL			
BOOT$L_UNIT	00018DDC-R	KERNEL	CONFIG		
BPTCODE	00000003	DEBUG			
BPTMAX	0000000F	DEBUG	EXCEPT		
BREAKUP_FILE	0002589F-R	PARSE	DIRECTORY	LOAD	RMS
BUG$CHECK	000271AF-R	QIO	FRKCTL	IOPOST	IOSNPG
			IOSRAM	LODMAP	MBAINT
			MEMALC	ONLY730	QIOREQ
CDB$BLOCK	00019118-R	CHAN730			
CHR$MCHK	00023E84-R	MCHK730	EXCEPT		
CLI_ACTION	0001D31A-R	CLI	APT		

Symbol	Value	Defined By	Referenced By ...		
CLK$RE_INIT	0001DFE4-R	CLOCK	CSDDBTDRV	KERNEL	SCB
CMK$ASTEXIT	0001D1A8-R	CHMK	ASTDEL		
CMK$CANTIM	0001DD88-R	CLOCK	CHMK		
CMK$HIBER	0001DD5D-R	CLOCK	CHMK		
CMK$INITSCB	00028203-R	SCB	CHMK		
CMK$MMENABLE	00024708-R	MEMMGT	CHMK		
CMK$RCONSOLE	0001E8C9-R	CONSOLE	CHMK	ONLY730	
CMK$REFRESH	0001AA29-R	KERNEL	CHMK		
CMK$SETIMR	0001DE21-R	CLOCK	CHMK		
CMK$SETIPL	00028431-R	SCB	CHMK		
CMK$SETPRT	00024728-R	MEMMGT	CHMK		
CMK$SGIPR	00020063-R	DEBUG	CHMK		
CMK$TCONSOLE	0001EA72-R	CONSOLE	CHMK	ONLY730	
COM$DELATTNAST	0002AE78-R	COMDRVSUB	ENTRY		
COM$DRVDEALMEM	0002AEEB-R	COMDRVSUB	ENTRY		
COM$FLUSHATTNS	0002AEAB-R	COMDRVSUB	ENTRY		
COM$POST	0002AED8-R	COMDRVSUB	ENTRY		
COM$SETATTNAST	0002AF10-R	COMDRVSUB	ENTRY		
COMMAND_TREE	000112A9-R	CLI	APT		
COND_DEBUG	0001FCB4-R	DEBUG	EXCEPT	KERNEL	
COND_HANDLR	00021964-R	EXCEPT	KERNEL		
CRD$EXPREG	000244C2-R	MEMMGT	CRDIMAGE		
CRD$GL_CRD_COMMAND_DEBUG	00019758-R	DSVECS	CLI		
CRD$K_CRD_INITIALIZATION	00000000	KERNEL			
CRD$K_DS_CONTROL_C	00000001	KERNEL			
CRD$K_DS_FLAGS	00000000	KERNEL			
CRD$K_HOOK_POINT	00000000	KERNEL			
CRD$K_LAST_ACCESS_CODE	00000002	KERNEL			
CRD$K_LAST_DS_VARIABLE	00000002	KERNEL			
CRD$K_LAST_FUNCTION	00000002	KERNEL			
CRD$K_LAST_HOOK_POINT	00000001	KERNEL			
CRD$K_READ	00000000	KERNEL			
CRD$K_TYPECODE	00000001	KERNEL			
CRD$K_WRITE	00000001	KERNEL			
CTL$GL_CCB	0001899C-R	KERNEL			
CTL$GL_CCBBASE	00018D9C-R	KERNEL	ASSIGN	IOPOST	IOSPGD
CTL$GW_CHINDX	000178C6-R	QIO	IOSPGD		
CTL$GW_NMIOCH	000178C4-R	QIO	IOSPGD		
CVT$T_MBAMAP	0001916C-R	CHAN730	PRINT		
CVT$T_MBASR	00019168-R	CHAN730	PRINT		
CVT$T_UBADPR	00019170-R	CHAN730	PRINT		
CVT$T_UBAMAP	00019174-R	CHAN730	PRINT		
DEBUG$GB_FLAGS	000197C4-R	DEBUG	CLI	EXCEPT	
DEBUG_HI	00019945-R	DEBUG	MEMMGT	SHOWMEMORY	
DEBUG_LO	00019941-R	DEBUG	MEMMGT	SHOWMEMORY	
DEBUG_THE_SUCKER	00019FE1-R	PRINT	SCRIPT		
DEF$C_BKSTP_LEN	00000011	RMS	DIRECTORY		
DEF$Q_BACKSTOP	00017C84-R	RMS	DIRECTORY		
DEF$Q_DEV	00019E6C-R	LOAD	DIRECTORY	KERNEL	RMS
DEF$Q_DIR	00019E92-R	LOAD	DIRECTORY	KERNEL	RMS
DEV$M_DMT	00200000	SYS$IODEF	DASSGN		
DEV$M_FOR	01000000	SYS$IODEF	DASSGN		

Symbol	Value	Defined By	Referenced By ...		
DEV$M_MNT	00080000	SYS$IODEF	DASSGN		
DEV$M_RCK	40000000	SYS$IODEF	DASSGN		
DEV$M_SWL	02000000	SYS$IODEF	DASSGN		
DEV$M_WCK	80000000	SYS$IODEF	DASSGN		
DEV$V_ALL	00000017	SYS$IODEF	DASSGN		
DEV$V_DMT	00000015	SYS$IODEF	DASSGN		
DEV$V_FOD	0000000E	SYS$IODEF	ASSIGN	QIOREQ	
DEV$V_FOR	00000018	SYS$IODEF	DASSGN	QIOREQ	
V$V_MBX	00000014	SYS$IODEF	ASSIGN		
V$V_MNT	00000013	SYS$IODEF	QIOREQ		
DEV$V_NET	0000000D	SYS$IODEF	ASSIGN		
DEV$V_SHR	00000010	SYS$IODEF	ASSIGN		
DEV$V_SPL	0000000C	SYS$IODEF	ASSIGN	QIOREQ	
DEV$V_SQD	00000005	SYS$IODEF	IOPOST		
DIAG_FILE_NAME	00019DF4-R	LOAD	DEBUG		
DISPATCH	00020C18-R	DISPAT	START		
DR$INITIAL	0002B5EB-R	DRINT	QIO		
DR$INT	0002B5D4-R	DRINT	QIO		
DS$AA_BPTADDR	000197D8-R	DEBUG	ERROR	EXCEPT	LOOP
DS$ABORT	00010020-R	ENTRY	EXCEPT	PARAM	SCRIPT
DS$ABORTWAIT	00010070-R	ENTRY			
DS$AB_BPTINST	000197C5-R	DEBUG	ERROR	LOOP	
DS$AL_DS_DISPATCH_VECTORS	0001975C-R	DSVECS	CRDIMAGE		
DS$AQ_SSEND	000107FF-R	ENTRY	KERNEL		
DS$AQ_SYSSRV	00010200-R	ENTRY	KERNEL	MEMMGT	
DS$ASKADR	00010090-R	ENTRY			
DS$ASKDATA	00010080-R	ENTRY			
DS$ASKLGCL	00010098-R	ENTRY			
DS$ASKSTR	000100A0-R	ENTRY			
DS$ASKVLD	00010088-R	ENTRY			
DS$ATTACH	000101A8-R	ENTRY			
DS$AX_ARB	0002C86C-R	KERNEL	SHOWMEMORY		
DS$AX_JIB	0002C800-R	KERNEL	SHOWMEMORY		
DS$AX_SOFTPCB	0002C8E4-R	KERNEL	ASSIGN	ASTCON	CANCEL
			CHMK	DASSGN	DEVALC
			EVENT	QIOREQ	SHOWMEMORY
DS$A_PRGBGN	00000200	KERNEL	APT	CLI	LOAD
			MEMMGT	SHOWMEMORY	
DS$BGNSUB	00010030-R	ENTRY			
DS$BRANCH	000100A8-R	ENTRY			
DS$BREAK	00010058-R	ENTRY			
DS$CANWAIT	00010070-R	ENTRY	CLOCK		
DS$CHANNEL	00010180-R	ENTRY	CONFIG	PROBE	
DS$CKLOOP	00010040-R	ENTRY			
DS$CLI	0001D1DD-R	CLI	DEBUG	EXCEPT	KERNEL
			SCRIPT		
DS$CLRVEC	00010168-R	ENTRY	START		
DS$CNTRLC	00010078-R	ENTRY	ATTACH	CRDIMAGE	DIRECTORY
			DISPAT	KERNEL	QAMAIN
DS$CVTREG	000100B0-R	ENTRY	CHAN730	DEBUG	EXCEPT
			FLAGS		
DS$DOSUMMARY	00010028-R	ENTRY			

Symbol	Value	Defined By	Referenced By ...		
DS$ENDPASS	00010010-R	ENTRY			
DS$ENDSUB	00010038-R	ENTRY			
DS$ENTRY	000293AD-R	VER730			
DS$ERRDEV	000100C8-R	ENTRY			
DS$ERRHARD	000100D0-R	ENTRY			
DS$ERRPREP	00010108-R	ENTRY			
DS$ERRSOFT	000100D8-R	ENTRY			
DS$ERRSYS	000100C0-R	ENTRY			
DS$ESCAPE	00010050-R	ENTRY			
DS$FAB_INPUT	00018600-R	KERNEL			
DS$FAB_OUTPUT	00018694-R	KERNEL			
DS$FREE_BGSYM	000101B8-R	ENTRY			
DS$GA_BREAKVEC	0001AC4C-R	SCB	EXCEPT		
DS$GA_BUFPTR	00018E4C-R	KERNEL			
DS$GA_CHKLPPC	00018E2C-R	KERNEL	DISPAT	ERROR	LOOP
DS$GA_CHMKVEC	0001A049-R	SCB	CHMK		
DS$GA_CRD_DS_INTERFACE	00019760-R	DSVECS	CLI	KERNEL	PRINT
			SCRIPT		
DS$GA_DS_CTRL_C_FIRST	00018728-R	KERNEL	CLI	CRDIMAGE	DSVECS
			QAMAIN		
DS$GA_DS_CTRL_C_SECOND	0001872C-R	KERNEL	CLI	CRDIMAGE	DSVECS
DS$GA_LASTADR	00018DF4-R	KERNEL	CRDIMAGE	MEMALC	MEMMGT
			SHOWMEMORY		
DS$GA_LOOPADR	00018E30-R	KERNEL	LOOP		
DS$GA_PTABLE	00019BF0-R	IOBASE	ATTACH	DEVICE	DSVECS
			KERNEL		
DS$GA_PTDESC	0001497F-R	IOBASE	ATTACH		
DS$GA_SELECTED	000199FC-R	IOBASE	CONFIG	DEVICE	
DS$GA_SELECTS	000199D8-R	IOBASE	ATTACH	DEVICE	KERNEL
DS$GA_SOFTINT	0001A059-R	SCB	CLOCK		
DS$GA_SUPPORT_TABLE	00017DC4-R	RMS			
DS$GA_TBITVEC	0001A051-R	SCB	EXCEPT		
DS$GA_USER_ERROR_TEXT	00019778-R	DSVECS	ERROR		
DS$GB_BYTEBUF	00018E50-R	KERNEL			
DS$GB_CI_INIT	00000000		WK-RMS		
DS$GB_INHIBIT_NAMING	00018E52-R	KERNEL	ATTACH	CLI	
DS$GB_MM_ENB	00019F4C-R	MEMMGT	CLI	DISPAT	KERNEL
DS$GB_QIOACT	0001A164-R	SCRIPT	PRINT		
DS$GB_STOPPING_THE_TIMER	0001A048-R	SCB	CLOCK		
DS$GB_TYPECODE	00018E51-R	KERNEL	DISPAT	ERROR	EXCEPT
			MCHK730	PRINT	SHOWMEMORY
			START		
DS$GB_UDA50_INIT	0002A0C4-R	PUBTDRIVR	RMS		
DS$GB_VDS_DEBUG	00019FE1-R	PRINT			
DS$GB_WIDTH	000196CA-R	CONSOLE	PRINT	QAMAIN	
DS$GETADDRESS	00010090-R	ENTRY			
DS$GETBUF	00010120-R	ENTRY			
DS$GETDATA	00010080-R	ENTRY			
DS$GETLOGICAL	00010098-R	ENTRY			
DS$GETSTRING	000100A0-R	ENTRY			
DS$GETTERM	000101C8-R	ENTRY			
DS$GETVIELD	00010088-R	ENTRY			

Symbol	Value	Defined By	Referenced By ...		
DS$GK_SID	03000000	ONLY730	KERNEL		
DS$GK_SUPPORT_TABLE_ROWS	00000017	RMS			
DS$GK_USOVR	00000200	ONLY730	CLOCK		
DS$GL_ACTUAL_MEMSIZE	00019F44-R	MEMMGT			
DS$GL_BUFCNT	00019100-R	BUFFER	KERNEL	MEMMGT	SHOWMEMORY
DS$GL_BUFLEN	00018E48-R	KERNEL	PARAM	PRINT	
DS$GL_CLIBASE	00019210-R	CLI	APT	ATTACH	CONFIG
			DEBUG	DIRECTORY	DISPAT
			KERNEL	LOAD	PARSE
			QACHECKS	QAMAIN	SCRIPT
			START		
DS$GL_CRD_DEBUG	0001978C-R	DSVECS	CLI	CRDIMAGE	
DS$GL_CRD_FLAGS	00019754-R	DSVECS	ATTACH	CONSOLE	CRDIMAGE
DS$GL_DEVERR_COUNT	00018E10-R	KERNEL	DISPAT	ERROR	
DS$GL_EFC0	00018DEC-R	KERNEL	FLAGS		
DS$GL_EFC1	00018DF0-R	KERNEL			
DS$GL_ERRCNT	00018E00-R	KERNEL	DISPAT	ERROR	LOOP
DS$GL_ERRSUP_COUNT	00018E18-R	KERNEL	DISPAT	ERROR	
DS$GL_FLAGS	00018DF8-R	KERNEL	APT	ATTACH	CALLFRAME
			CLI	CLOCK	CONSOLE
			CRDIMAGE	CSDDBTDRV	DISPAT
			DSVECS	ERROR	EXCEPT
			LOAD	LOOP	MEMMGT
			PRINT	SCB	SCRIPT
			SHOWMEMORY	START	
DS$GL_FSTTEST	00018E20-R	KERNEL	DISPAT	QACHECKS	QAMAIN
			START		
DS$GL_HARDERR_COUNT	00018E04-R	KERNEL	DISPAT	ERROR	
DS$GL_LOOKASIDE	00019F24-R	MEMALC	SHOWMEMORY		
DS$GL_LSTTEST	00018E24-R	KERNEL	DISPAT	LOOP	QACHECKS
			QAMAIN	START	
DS$GL_MEMSIZE	00019F40-R	MEMMGT	DEBUG	MEMALC	SHOWMEMORY
DS$GL_NUMTEST	00018E1C-R	KERNEL	START		
DS$GL_PAGE	00019F69-R	PRINT			
DS$GL_PCBVEC	00018DA0-R	KERNEL	ENTRY		
DS$GL_PREPERR_COUNT	00018E08-R	KERNEL	ERROR		
DS$GL_RADIX	00018E40-R	KERNEL	DEBUG	PARSE	
DS$GL_RUNTIM	00018E34-R	KERNEL			
DS$GL_SET_MEMSIZE	00019F48-R	MEMMGT	SHOWMEMORY		
DS$GL_SIZE	00018E44-R	KERNEL			
DS$GL_SOFTERR_COUNT	00018E0C-R	KERNEL	DISPAT	ERROR	
DS$GL_SUBTEST	00018E28-R	KERNEL	DISPAT	LOOP	QACHECKS
			START		
DS$GL_SYSERR_COUNT	00018E14-R	KERNEL	DISPAT	ERROR	
DS$GL_TERMCHAR	000190F?-R	KERNEL	CONSOLE		
DS$GL_TIMESEC	00018L?8-R	KERNEL			
DS$GL_WAITIM	00018E3C-R	KERNEL			
DS$GPHARD	00010018-R	ENTRY	CHAN730	DEVICE	ERROR
			QIO	START	
DS$GQ_BUFQWD	00018E48-R	KERNEL			
DS$GQ_CURYEAR	00019668-R	CLOCK	KERNEL		
DS$GQ_DAYTIM	00018560-R	VER730	KERNEL		

Symbol	Value	Defined By	Referenced By ...		
DS$GQ_EFL	00018DEC-R	KERNEL	EVENT		
DS$GQ_PHYADR	000190F8-R	BUFFER	MEMMGT		
DS$GQ_TESTID	0001996C-R	ERROR	START		
DS$GT_ASCIBUF	00018E53-R	KERNEL			
DS$GT_BUFFER	00018E62-R	KERNEL	PRINT	SCRIPT	
DS$GT_NEWYEAR	00019670-R	CLOCK	KERNEL		
DS$GT_PROMPT	000185BD-R	VER730	CLI	KERNEL	
DS$GT_START	0001857F-R	VER730	KERNEL		
DS$GT_STRBUF	00019062-R	KERNEL	FLAGS	PARAM	
DS$GT_VALPHA	000256A0-R	PARSE			
DS$GT_VALPHANUM	000256B2-R	PARSE			
DS$GT_VDS_VERSION	000185B4-R	VER730	ERROR		
DS$GT_VFILE	000256C4-R	PARSE			
DS$GT_VHEX	00025725-R	PARSE			
DS$GT_VSPACE	000256D6-R	PARSE			
DS$GT_VSYMBOL	0002568E-R	PARSE			
DS$GW_TTIN	00018DFC-R	KERNEL	PRINT	SCRIPT	
DS$GW_TTOUT	00018DFE-R	KERNEL	CONSOLE	ERROR	EXCEPT
			PRINT	SCRIPT	
DS$HELP	000101B0-R	ENTRY			
DS$INITSCB	00010170-R	ENTRY	DISPAT	KERNEL	
DS$INLOOP	00010048-R	ENTRY			
DS$K_ASCIBUF	0000000F	KERNEL			
DS$K_BBC	0000001D	DISPAT			
DS$K_BBCC	00000021	DISPAT			
DS$K_BBCCI	00000023	DISPAT			
DS$K_BBCS	0000001F	DISPAT			
DS$K_BBS	0000001C	DISPAT			
DS$K_BBSC	00000020	DISPAT			
DS$K_BBSS	0000001E	DISPAT			
DS$K_BBSSI	00000022	DISPAT			
DS$K_BCC_M	0000001B	DISPAT			
DS$K_BCS_M	0000001A	DISPAT			
DS$K_BEQLU_B	00000005	DISPAT			
DS$K_BEQLU_M	00000011	DISPAT			
DS$K_BEQL_B	00000004	DISPAT			
DS$K_BEQL_M	00000010	DISPAT			
DS$K_BERROR	00000026	DISPAT			
DS$K_BGEQU_B	00000007	DISPAT			
DS$K_BGEQU_M	00000013	DISPAT			
DS$K_BGEQ_B	00000006	DISPAT			
DS$K_BGEQ_M	00000012	DISPAT			
DS$K_BGTRU_B	00000009	DISPAT			
DS$K_BGTRU_M	00000015	DISPAT			
DS$K_BGTR_B	00000008	DISPAT			
DS$K_BGTR_M	00000014	DISPAT			
DS$K_BLBC	00000025	DISPAT			
DS$K_BLBS	00000024	DISPAT			
DS$K_BLEQU_B	00000003	DISPAT			
DS$K_BLEQU_M	0000000F	DISPAT			
DS$K_BLEQ_B	00000002	DISPAT			
DS$K_BLEQ_M	0000000E	DISPAT			

Symbol	Value	Defined By	Referenced By ...		
DS$K_BLSSU_B	00000001	DISPAT			
DS$K_BLSSU_M	0000000D	DISPAT			
DS$K_BLSS_B	00000000	DISPAT			
DS$K_BLSS_M	0000000C	DISPAT			
DS$K_BNEQU_B	0000000B	DISPAT			
DS$K_BNEQU_M	00000017	DISPAT			
DS$K_BNEQ_B	0000000A	DISPAT			
DS$K_BNEQ_M	00000016	DISPAT			
DS$K_BNERROR	00000027	DISPAT			
DS$K_BUFSIZ	00000200	KERNEL	PRINT	SCRIPT	
DS$K_BVC_M	00000019	DISPAT			
DS$K_BVS_M	00000018	DISPAT			
DS$K_CCB	0000004C	KERNEL	QIO		
DS$K_NYSIZ	00000014	CLOCK	KERNEL		
DS$K_PRGSIZ	0000F800	KERNEL	APT	LOAD	MEMMGT
DS$K_SSSIZE	00000600	ENTRY	KERNEL		
DS$K_STRBUF	00000080	KERNEL	FLAGS	PARAM	
DS$LOAD	00010198-R	ENTRY	CRDIMAGE	QIO	SCRIPT
DS$LOADPCS	000101C0-R	ENTRY	KERNEL		
DS$L_USERCNTRLC	00018730-R	KERNEL	CLI	CRDIMAGE	
DS$MAPDBGBLOCK	00010118-R	ENTRY			
DS$MMOFF	00010158-R	ENTRY	CRDIMAGE		
DS$MMON	00010150-R	ENTRY			
DS$PARSE	000100B8-R	ENTRY			
DS$PRINTB	000100E0-R	ENTRY	CHAN730		
DS$PRINTF	000100F0-R	ENTRY	ATTACH	DEBUG	DEVICE
			DIRECTORY	DSLOAD	HELP
			QACHECKS	QAFLAGS	QAMAIN
			SHOWMEMORY		
DS$PRINTS	000100F8-R	ENTRY			
DS$PRINTSIG	00010100-R	ENTRY			
DS$PRINTX	000100E8-R	ENTRY			
DS$PROBE	000101A0-R	ENTRY			
DS$RAB_INPUT	00018650-R	KERNEL	SCRIPT		
DS$RAB_OUTPUT	000186E4-R	KERNEL	PRINT	SCRIPT	
DS$RELBUF	00010128-R	ENTRY			
DS$SERVICE_DISPATCHER	000101D0-R	ENTRY			
DS$SETIPL	00010178-R	ENTRY			
DS$SETMAP	00010188-R	ENTRY			
DS$SETPRIEXV	00010110-R	ENTRY			
DS$SETVEC	00010160-R	ENTRY			
DS$SHOCHAN	00010190-R	ENTRY			
DS$SHOWCHAN	00010190-R	ENTRY			
DS$SOFTIPL5	0001A402-R	KERNEL	SCB		
DS$SRVDISP_TABLE	000137B0-R	ENTRY			
DS$SUMMARY	00010028-R	ENTRY	DISPAT	LOOP	
DS$USERMODE	0001A40D-R	KERNEL	VER730		
DS$WAITMS	00010060-R	ENTRY			
DS$WAITUS	00010068-R	ENTRY	CLOCK	CONFIG	ONLY730
DSA$M_DFLTFLGS	00003000	KERNEL	FLAGS		
DSI$CHANNEL_VEC	0001D0EC-R	CHAN730	SCB		
DSI$TIMER	0001DEF4-R	CLOCK	SCB		

Symbol	Value	Defined By	Referenced By ...		
DSI$TIMSRV	0001DE94-R	CLOCK	SCB		
DSP$CONFIG_ADP	00024D4C-R	ONLY730	CONFIG		
DSP$CONFIG_LOAD	0001E0B3-R	CONFIG			
DSP$GEN_PTABLES	0002023B-R	DEVICE	START		
DSP$REMOVEBREAK	0001FC29-R	DEBUG			
DSP$SHOWMBA	0001CFED-R	CHAN730			
DSP$SHOWUBI	0001CFF7-R	CHAN730			
DSR$ALLOCATE_PSEUDO_RPB	000275CC-R	RMS	CONFIG	DIRECTORY	
DSR$CHECKLOAD	00028A4C-R	SHOWMEMORY	CALLFRAME	DISPAT	MEMMGT
DSR$CHECK_AUTOTEST_OFF_SET	0001AD04-R	KERNEL	CRDIMAGE	DSVECS	ERROR
DSR$CHECK_MENUTEST_OFF_SET	0001ACE0-R	KERNEL	CRDIMAGE	DSVECS	ERROR
DSR$COMPLETION	000217E2-R	ERROR	ATTACH	CLI	DEBUG
			DIRECTORY	LOAD	QIO
			SCRIPT		
DSR$DEALLOCATE_PSEUDO_RPB	00027645-R	RMS	CONFIG	DIRECTORY	
DSR$FAULT_CLEAR	00023EE8-R	MCHK730	EXCEPT	SCB	
DSR$FIND_USER_PC	0001CD3B-R	CALLFRAME	DISPAT	EXCEPT	START
DSR$IFLOADDEV	00023AAE-R	LOAD	ATTACH		
DSR$KEY_EQUAL	000234E9-R	HELP	DIRECTORY		
DSR$LOAD_CRD	0001ED28-R	CRDIMAGE	CLI	KERNEL	
DSR$MMENABLE	000246F8-R	MEMMGT	DISPAT	SHOWMEMORY	
DSR$PRINT_TYPECODE_NAME	0001F4AE-R	CRDIMAGE			
DSR$RESTORE_CRD_VECTORS	0001F036-R	CRDIMAGE	KERNEL		
DSR$SETIMR_INI	0001DDC0-R	CLOCK	KERNEL	MEMMGT	
DSR$UNLOAD_CRD	0001EF53-R	CRDIMAGE	CLI	KERNEL	
DSV$ATTACH	0001BA6C-R	ATTACH	CLI		
DSV$DEATTACH	0001C5A3-R	ATTACH	CLI		
DSV$DEBUG	000200A1-R	DEBUG	CLI	START	
DSV$DESELECT	0001C7C2-R	ATTACH	CLI		
DSV$DIRECTORY	00020BEF-R	DIRECTORY	CLI		
DSV$EXIT	0001AC37-R	KERNEL	CLI	CRDIMAGE	DSVECS
DSV$INITPCS	00025942-R	PCSLOAD	CLI		
DSV$LOAD	00023AD9-R	LOAD	CLI		
DSV$RUN	00023AD8-R	LOAD	CLI		
DSV$SELECT	0001C63C-R	ATTACH	CLI		
DSV$SETLOAD	00023A12-R	LOAD	CLI	CONFIG	KERNEL
DSV$SETPAGE	00025952-R	PRINT	CLI		
DSV$SHOWDEVICE	0001C511-R	ATTACH	CLI	QAMAIN	
DSV$SHOWDEVICEB	0001C50A-R	ATTACH	CLI		
DSV$SHOWLOAD	00023A8E-R	LOAD	CLI		
DSV$SHOWMEMORY	00028A6B-R	SHOWMEMORY	CLI		
DSV$SHOWPAGE	00025979-R	PRINT	CLI		
DSV$SHOWSELECT	0001C572-R	ATTACH	CLI	QAMAIN	
DSV$SHOWSELECTB	0001C56B-R	ATTACH	CLI		
DSX$ABORT	00029228-R	START	ENTRY	QACHECKS	QAMAIN
DSX$ATTACH	0001BB76-R	ATTACH	ENTRY		
DSX$BGNSUB	00023C0E-R	LOOP	ENTRY		
DSX$BRANCH	00026560-R	QAMAIN	ENTRY		
DSX$CANWAIT	0001DC0C-R	CLOCK	ENTRY		
DSX$CHANNEL	0001CD6C-R	CHAN730	ENTRY		
DSX$CHECK_SUPPORT	00027658-R	RMS			
DSX$CKLOOP	00023E1D-R	LOOP	ENTRY		

Symbol	Value	Defined By	Referenced By ...		
DSX$CLRVEC	00028372-R	SCB	ENTRY		
DSX$CNTRLC	0001AC0C-R	KERNEL	ENTRY		
DSX$CVTREG	00025DD4-R	PRINT	ENTRY		
DSX$DIRECTORY	00020A01-R	DIRECTORY			
DSX$DISMOUNT	0002B5B4-R	DEVALC	ENTRY		
DSX$DOSUMMARY	00021045-R	DISPAT	ENTRY		
DSX$ENDPASS	00020EB8-R	DISPAT	ENTRY		
DSX$ENDSUB	00023CAC-R	LOOP	ENTRY		
DSX$ERRDEV	0002144A-R	ERROR	ENTRY		
DSX$ERRHARD	000213FC-R	ERROR	ENTRY		
DSX$ERRPREP	00021423-R	ERROR	ENTRY		
DSX$ERRSOFT	000213D5-R	ERROR	ENTRY		
DSX$ERRSYS	00021478-R	ERROR	ENTRY		
DSX$ESCAPE	00023BF9-R	LOOP	ENTRY		
DSX$FREEDBGSYM	0002489F-R	MEMMGT	ENTRY		
DSX$GETADDRESS	000250A8-R	PARAM	ENTRY		
DSX$GETBUF	0001CBBC-R	BUFFER	ENTRY		
DSX$GETDATA	00024FD0-R	PARAM	ENTRY		
DSX$GETLOGICAL	000250F2-R	PARAM	ENTRY		
DSX$GETSTRING	000251C4-R	PARAM	ENTRY		
DSX$GETTERM	0001AD23-R	KERNEL	ENTRY		
DSX$GETVIELD	00025046-R	PARAM	ENTRY		
DSX$GPHARD	00020209-R	DEVICE	ENTRY		
DSX$HELP	00022D44-R	HELP	CLI	ENTRY	
DSX$INITSCB	000281E0-R	SCB	ENTRY		
DSX$INLOOP	00023E5D-R	LOOP	ENTRY		
DSX$LOAD	000211A0-R	DSLOAD	DEBUG	ENTRY	LOAD
DSX$LOADPCS	00025938-R	PCSLOAD	ENTRY		
DSX$MAPDBGBLOCK	00024851-R	MEMMGT	ENTRY		
DSX$MMOFF	000246DE-R	MEMMGT	CLI	ENTRY	
DSX$MMON	000246D0-R	MEMMGT	CLI	ENTRY	
DSX$MOUNT	0002B59A-R	DEVALC	ENTRY		
DSX$PARSE	00025401-R	PARSE	ENTRY		
DSX$PRINT	00025C0D-R	PRINT	ATTACH	CALLFRAME	CLI
			CONSOLE	CRDIMAGE	DISPAT
			DSVECS	ERROR	FLAGS
			KERNEL	LOAD	LOOP
			MEMMGT	PARAM	QIO
			SCRIPT	START	
DSX$PRINTB	00025C75-R	PRINT	ENTRY	EXCEPT	
DSX$PRINTF	00025C8D-R	PRINT	ENTRY	LOAD	
DSX$PRINTI	00025CA2-R	PRINT	DEBUG	EXCEPT	KERNEL
			PCSLOAD		
DSX$PRINTS	00025C5B-R	PRINT	ENTRY		
DSX$PRINTSIG	00021A6C-R	EXCEPT	ENTRY		
DSX$PRINTSIGI	00021A7E-R	EXCEPT			
DSX$PRINTX	00025C68-R	PRINT	ENTRY	EXCEPT	
DSX$PROBE	0002BED8-R	PROBE	ENTRY		
DSX$RELBUF	0001CC0E-R	BUFFER	ENTRY		
DSX$SERVICE_DISPATCHER	0002139C-R	ENTRY			
DSX$SETIPL	00028418-R	SCB	ENTRY		
DSX$SETMAP	0001CE1D-R	CHAN730	ENTRY		

Symbol	Value	Defined By	Referenced By ...		
DSX$SETPRIEXV	00022021-R	EXCEPT	ENTRY		
DSX$SETVEC	000282B7-R	SCB	ENTRY		
DSX$SHOWCHAN	0001D008-R	CHAN730	ENTRY		
DSX$SUPERPARSE	000254BA-R	PARSE	APT	CLI	
DSX$TYPE_OUT	00025999-R	PRINT	SCRIPT		
DSX$WAITMS	0001DC33-R	CLOCK	ENTRY		
DSX$WAITUS	0001DCBC-R	CLOCK	ENTRY		
DSX$WAITUS_USOVR	0001DCD7-R	CLOCK	KERNEL		
DS_CLEANUP	00020F82-R	DISPAT	CLI	CRDIMAGE	DIRECTORY
			KERNEL	LOAD	LOOP
			SHOWMEMORY	START	
DS_ERRSUP	000216B5-R	ERROR	APT	ASTDEL	ATTACH
			CHMK	CLI	CLOCK
			CONFIG	DEVICE	DISPAT
			ENTRY	EXCEPT	IOPOST
			KERNEL	LOOP	MEMALC
			MEMMGT	ONLY730	QIO
			SCRIPT		
DS_GETLINE	00028684-R	SCRIPT	PARAM		
DS_SUPERLINE	0002867D-R	SCRIPT	ATTACH	CLI	
DS_TESTID	00021649-R	ERROR	START		
DYN$C_ACB	00000002	QIO			
DYN$C_ADP	00000001	QIO			
DYN$C_AQB	00000003	QIO			
DYN$C_BRDCST	0000001A	QIO			
DYN$C_BUFIO	00000013	QIO			
DYN$C_CEB	00000004	QIO			
DYN$C_CRB	00000005	QIO			
DYN$C_CXB	0000001B	QIO			
DYN$C_DDB	00000006	QIO			
DYN$C_DPT	0000001E	QIO			
DYN$C_EXTGSD	00000028	QIO			
DYN$C_FCB	00000007	QIO			
DYN$C_FRK	00000008	QIO			
DYN$C_GSD	00000015	QIO			
DYN$C_IDB	00000009	QIO			
DYN$C_IRP	0000000A	QIO			
DYN$C_IRPE	0000002C	QIO			
DYN$C_JIB	0000002F	QIO			
DYN$C_JPB	0000001F	QIO			
DYN$C_KFH	00000026	QIO			
DYN$C_KFI	00000018	QIO			
DYN$C_LOG	0000000B	QIO			
DYN$C_MBX	0000002B	QIO			
DYN$C_MTL	00000019	QIO			
DYN$C_MVL	00000016	QIO			
DYN$C_NDB	0000001C	QIO			
DYN$C_NET	00000017	QIO			
DYN$C_PBH	00000020	QIO			
DYN$C_PCB	0000000C	QIO			
DYN$C_PDB	00000021	QIO			
DYN$C_PFL	00000023	QIO			

Symbol	Value	Defined By	Referenced By ...		
DYN$C_PIB	00000022	QIO			
DYN$C_PQB	0000000D	QIO			
DYN$C_PTR	00000025	QIO			
DYN$C_RBM	00000031	QIO			
DYN$C_RVT	0000000E	QIO			
DYN$C_RVX	00000027	QIO			
DYN$C_SFT	00000024	QIO			
DYN$C_SHB	0000002A	QIO			
DYN$C_SHMCEB	0000002E	QIO			
DYN$C_SHMGSD	00000029	QIO			
DYN$C_SHRBUFIO	00000080	QIO			
DYN$C_SLAVCEB	0000002D	QIO			
DYN$C_SPECIAL	0000008C	QIO			
DYN$C_SSB	0000001D	QIO			
DYN$C_TQE	0000000F	QIO			
DYN$C_TWP	00000030	QIO			
DYN$C_TYPAHD	00000014	QIO			
DYN$C_UCB	00000010	QIO			
DYN$C_VCA	00000032	QIO			
DYN$C_VCB	00000011	QIO			
DYN$C_WCB	00000012	QIO			
ERL$DEVICERR	000293B7-R	VMSDUMMY	ENTRY		
ERL$DEVICTMO	000293B7-R	VMSDUMMY	ENTRY		
ERL$RELEASEMB	000293B7-R	VMSDUMMY	ENTRY	IOSNPG	
ESTKPTR	0002FE00-R	KERNEL	EXCEPT	MEMMGT	SHOWMEMORY
EXCHANGE_DISPATCH_VECTORS	0001F081-R	CRDIMAGE			
EXE$ABORTIO	0002C30A-R	QIOREQ	ACPFDT QIOFDT	COMDRVSUB	ENTRY
EXE$ACVIOLAT	00021DE4-R	EXCEPT			
EXE$ALLOC	0002B446-R	DEVALC	ENTRY		
EXE$ALLOCATE	00023FB6-R	MEMALC			
EXE$ALLOCBUF	00023EEF-R	MEMALC	COMDRVSUB	ENTRY	QIOREQ
EXE$ALLOCCEB	00023EF3-R	MEMALC			
EXE$ALLOCIRP	00023F00-R	MEMALC	ENTRY	QIOREQ	
EXE$ALLOCJIB	00023EF7-R	MEMALC			
EXE$ALLOCPCB	00023F04-R	MEMALC			
EXE$ALLOCPQB	00023F0C-R	MEMALC			
EXE$ALLOCTQE	00023F15-R	MEMALC	CLOCK		
EXE$ALONONPAGED	00023F4F-R	MEMALC	ATTACH CRDIMAGE ODS QIO	CANCEL DSVECS ONLY730 RMS	CONFIG ENTRY QAMAIN SCRIPT
EXE$ALTQUEPKT	0002C335-R	QIOREQ	ENTRY		
EXE$ASCTIM	0002AFD5-R	CVTTIM	ENTRY		
EXE$ASSIGN	0002ABDC-R	ASSIGN	ENTRY		
EXE$BINTIM	0002B076-R	CVTTIM	ENTRY		
EXE$BREAK	00021E0C-R	EXCEPT			
EXE$BUFFRQUOTA	00029388-R	VMSDUMMY	ENTRY		
EXE$BUFQUOPRC	00029388-R	VMSDUMMY	ENTRY		
EXE$CANCEL	0002AD73-R	CANCEL	ENTRY		
EXE$CANTIM	0001DD73-R	CLOCK	ENTRY		
EXE$CARRIAGE	0002C127-R	QIOFDT	ENTRY		

Symbol	Value	Defined By	Referenced By ...		
EXE$CLREF	000218A9-R	EVENT	ENTRY		
EXE$CNTREG	000245E3-R	MEMMGT	ENTRY		
EXE$DALLOC	0002B4DF-R	DEVALC			
EXE$DASSGN	0002B386-R	DASSGN	ENTRY		
EXE$DEALLOCATE	00024051-R	MEMALC			
EXE$DEANONPAGED	00023FDE-R	MEMALC	ASTDEL	ATTACH	CLOCK
			COMDRVSUB	CONFIG	CRDIMAGE
			DASSGN	DSVECS	ENTRY
			IOPOST	ONLY730	QAMAIN
			QIO	QIOFDT	RMS
			SCRIPT		
EXE$EXPREG	000244DF-R	MEMMGT	ENTRY		
EXE$FAO	0002206C-R	FAO	ENTRY		
EXE$FAOL	00022068-R	FAO	ENTRY		
EXE$FINISHIO	0002C319-R	QIOREQ	ENTRY	IOSRAM	QIOFDT
EXE$FINISHIOC	0002C317-R	QIOREQ	ACPFDT	ENTRY	
EXE$FORK	0002B5F9-R	FRKCTL	COMDRVSUB	ENTRY	
EXE$FORKDSPTH	00028618-R	FRKCTL	QIO	SCB	
EXE$GB_ADPTYPE	00029715-R	BOOTDRIVR			
EXE$GB_CPUTYPE	00029714-R	BOOTDRIVR	CSDDBTDRV	DISPAT	KERNEL
EXE$GETTIM	0002B65C-R	GETTIM	ENTRY		
EXE$GL_ABSTIM	00010810-R	ENTRY	CLOCK	IOSNPG	
EXE$GL_PWRDONE	00017B90-R	QIO			
EXE$GL_SCB	0001A055-R	SCB			
EXE$GL_SPLITADR	00019F20-R	MEMALC	SHOWMEMORY		
EXE$GQ_SYSTIME	00010B00-R	ENTRY	CLOCK	CVRTIM	GETTIM
			IOSNPG	KERNEL	
EXE$GTCHAN	0002B678-R	GTCHAN	ENTRY		
EXE$HIBER	0001DD3D-R	CLOCK	ENTRY		
EXE$INSERTIRP	0002C38D-R	QIOREQ	ENTRY		
EXE$INSIOQ	0002C372-R	QIOREQ	ENTRY	IOPOST	
EXE$INSTIMQ	0001DDDD-R	CLOCK			
EXE$IOFORK	0002B5F4-R	FRKCTL	ENTRY		
EXE$MAXACMODE	000271A0-R	QIO	ASSIGN	COMDRVSUB	DEVALC
			ENTRY		
EXE$MODIFY	0002BF9C-R	QIOFDT	ENTRY		
EXE$MODIFYLOCK	0002BFD4-R	QIOFDT	ENTRY		
EXE$MODIFYLOCKR	0002BFD7-R	QIOFDT	ENTRY		
EXE$NUMTIM	0002B26C-R	CVRTIM	ENTRY		
EXE$ONEPARM	0002BF90-R	QIOFDT	ENTRY		
EXE$PWRTIMCHK	000275B9-R	QIO	ENTRY		
EXE$QIO	00026B94-R	QIO	ENTRY		
EXE$QIOACPPKT	0002C357-R	QIOREQ	CANCEL		
EXE$QIODRVPKT	0002C331-R	QIOREQ	ACPFDT	ENTRY	QIOFDT
EXE$QIORETURN	C002C369-R	QIOREQ	ENTRY		
EXE$READ	0002BFA2-R	QIOFDT	ENTRY		
EXE$READCHK	0002C05E-R	QIOFDT	ENTRY		
EXE$READCHKR	0002C072-R	QIOFDT	ENTRY		
EXE$READEF	000218D6-R	EVENT	ENTRY		
EXE$READLOCK	0002BFCE-R	QIOFDT	ACPFDT	ENTRY	
EXE$READLOCKR	0002BFE1-R	QIOFDT	ENTRY		
EXE$ROPRAND	00021DD4-R	EXCEPT			

Symbol	Value	Defined By	Referenced By ...		
EXE$SENSEMODE	0002C0FF-R	QIOFDT	ENTRY		
EXE$SETAST	0001B8A0-R	ASTCON	ENTRY	VMSDUMMY	
EXE$SETCHAR	0002C0D1-R	QIOFDT	ENTRY		
EXE$SETEF	0002188F-R	EVENT	ENTRY		
EXE$SETIMR	0001DE0F-R	CLOCK	ENTRY		
EXE$SETMODE	0002C0EF-R	QIOFDT	ENTRY		
EXE$SETPRT	00024719-R	MEMMGT	ENTRY		
EXE$SNDEVMSG	000293B7-R	VMSDUMMY	ENTRY		
EXE$TBIT	00021E38-R	EXCEPT			
EXE$UNWIND	0002C448-R	UNWIND	ENTRY		
EXE$WAITFR	000218FA-R	EVENT	ENTRY		
EXE$WAKE	0001DC27-R	CLOCK	ENTRY		
EXE$WFLAND	00021938-R	EVENT	ENTRY		
EXE$WFLOR	00021917-R	EVENT	ENTRY		
EXE$WRITE	0002BFAB-R	QIOFDT	ENTRY		
EXE$WRITECHK	0002C064-R	QIOFDT	ENTRY		
EXE$WRITECHKR	0002C09C-R	QIOFDT	ENTRY		
EXE$WRITELOCK	0002BFD1-R	QIOFDT	ACPFDT	ENTRY	
EXE$WRITELOCKR	0002BFE8-R	QIOFDT	ENTRY		
EXE$ZEROPARM	0002BF96-R	QIOFDT	ENTRY		
EXE_ACCVIO	00021DE4-R	EXCEPT	SCB		
EXE_ARITH	00021E00-R	EXCEPT	SCB		
EXE_BREAK	00021E0C-R	EXCEPT	SCB		
EXE_CHME	00021E6C-R	EXCEPT	SCB		
EXE_CHMK	00021E64-R	EXCEPT			
EXE_CHMS	00021E74-R	EXCEPT	SCB		
EXE_CHMU	00021E7C-R	EXCEPT	SCB		
EXE_CMODKRNL	0001D154-R	CHMK	SCB		
EXE_COMPAT	00021DF8-R	EXCEPT	SCB		
EXE_KRNLSTK	00021DAC-R	EXCEPT	SCB		
EXE_MCHK	00021D8C-R	EXCEPT	SCB		
EXE_OPCCUS	00021DCC-R	EXCEPT	SCB		
EXE_OPCDEC	00021DC4-R	EXCEPT	SCB		
EXE_POWER	00021DB8-R	EXCEPT	SCB		
EXE_RADRMOD	00021DDC-R	EXCEPT	SCB		
EXE_ROPRAND	00021DD4-R	EXCEPT	SCB		
EXE_TBIT	00021E38-R	EXCEPT	SCB		
EXE_TRANSL	00021DEC-R	EXCEPT	SCB		
EXE_UNEXPINT	00021E59-R	EXCEPT	ONLY730	SCB	
FETCH_LONG	00024E76-R	ONLY730			
FETCH_STORE	0001FE99-R	DEBUG	ONLY730		
FETCH_STOR_PREG	00020072-R	DEBUG			
FIL$CACHE_INIT	00022602-R	FILEREAD			
FIL$CACHE_TRUNC	00022676-R	FILEREAD			
FIL$CVTFILNAM	0001F600-R	CVTFILNAM	FILEREAD	RMS	RT11
FIL$C_DIR_SIZE	00000024	FILEREAD			
FIL$C_SIZE	00000218	FILEREAD			
FIL$FINDFILID	000227DD-R	FILEREAD			
FIL$GQ_CACHE	00000000		WK-FILEREAD		
FIL$GT_DDDEV	00016BB3-R	LOAD	WK-FILEREAD		
FIL$GT_DDSTRING	00016BB4-R	LOAD	FILEREAD		
FIL$GT_TOPSYS	00000000		WK-FILEREAD		

Symbol	Value	Defined By	Referenced By ...		
FIL$MOUNT	0002274D-R	FILEREAD			
FIL$OPENFILF	000224FC-R	FILEREAD	ODS		
FIL$RDCHKFILHDR	00022AE4-R	FILEREAD			
FIL$RDWRTLBN	0001CB68-R	BOOTIO	FILEREAD		
FIL$READVBN	00022BC3-R	FILEREAD	ODS		
FIL$STATBLK	00022C7A-R	FILEREAD			
FIL$WRITEVBN	00022BBC-R	FILEREAD			
FIND_USERPC	00021D31-R	EXCEPT			
GET_BLOCK_LENGTH	0001B6A0-R	ANSI	ANSI		
GT_MENU_ERROR	0001121C-R	CLI	KERNEL		
HELPINFO	000199A8-R	HELP	CLI		
HP$C_NAMELEN	00000014	ATTACH			
IMAGE_HEADER	0001879C-R	KERNEL	DEBUG	DSLOAD	LOAD
IMAGE_HEADER_END	0001899C-R	KERNEL	DSLOAD	LOAD	
INI$BRK	0001A400-R	KERNEL	CLI		
INI$LOAD_DEVICE	0001E034-R	CONFIG	KERNEL	SHOWMEMORY	
INI$PTABLE	0001C92E-R	ATTACH	KERNEL	SHOWMEMORY	
INI$RDONLY	0002497C-R	MEMMGT			
INI$WRITABLE	0002497C-R	MEMMGT			
INIT_CONTEXT	0001A93B-R	KERNEL	CLI	DIRECTORY	DISPAT
			LOAD	LOOP	SCRIPT
			START		
INS$PTABLE	0001CAD0-R	ATTACH	CONFIG	ONLY730	
IO$GQ_PHYSICAL	00017134-R	ONLY730	MEMMGT	PROBE	
IO$M_CANCTRLO	00000040	SYS$IODEF	EXCEPT		
IO$M_CTRLCAST	00000100	SYS$IODEF	KERNEL	PRINT	
IO$M_FCODE	0000003F	SYS$IODEF	QIOREQ		
IO$M_NOW	00000040	SYS$IODEF	KERNEL		
IO$M_TRMNOECHO	00001000	SYS$IODEF	PRINT		
IO$TESTCSR_L	00024ECD-R	ONLY730	QIO		
IO$TESTCSR_W	00024EEC-R	ONLY730	QIO		
IO$V_CANCTRLO	00000006	SYS$IODEF	CONSOLE		
IO$_LOGICAL	0000002F	SYS$IODEF	QIOREQ		
IO$_PHYSICAL	0000001F	SYS$IODEF	QIOREQ		
IO$_READLBLK	00000021	SYS$IODEF	QIOREQ		
IO$_READPROMPT	00000037	SYS$IODEF	CONSOLE	SCRIPT	
IO$_READVBLK	00000031	SYS$IODEF	KERNEL	PRINT	QIOREQ
			SCRIPT		
IO$_SENSEMODE	00000027	SYS$IODEF	CONSOLE		
IO$_SETMODE	00000023	SYS$IODEF	CONSOLE	KERNEL	PRINT
IO$_WRITEVBLK	00000030	SYS$IODEF	ERROR	EXCEPT	PRINT
			QIOREQ	SCRIPT	
IOC$ALLOSPT	0002BA0C-R	IOSNPG	ENTRY		
IOC$ALOUBAMAP	0002B963-R	IOSNPG	ENTRY		
IOC$ALOUBAMAPN	0002B95F-R	IOSNPG	ENTRY		
IOC$ALTUBAMAP	0002B941-R	IOSNPG	ENTRY		
IOC$AL_SYSUCB	0002B700-R	IOBASE			
IOC$APPLYECC	0002BBAC-R	IOSRAM	ENTRY		
IOC$CANCELIO	00028718-R	IOSNPG	ENTRY		
IOC$CVTLOGPHY	0002BC20-R	IOSRAM	ACPFDT	IOPOST	
IOC$CVT_DEVNAM	0002BA21-R	IOSPGD	DEVALC		
IOC$DIAGBUFILL	0002B72F-R	IOSNPG	ENTRY		

Symbol	Value	Defined By	Referenced By ...		
IOC$DIRPOST1	00023952-R	IOPOST			
IOC$FFCHAN	0002BA55-R	IOSPGD	ASSIGN		
IOC$FILSPT	0002AD61-R	BUFCTL			
IOC$GETBYTE	0002AD08-R	BUFCTL	ENTRY		
IOC$GL_ADPLIST	0002B70C-R	IOBASE	ONLY730	QIO	
IOC$GL_DEVLIST	0002B708-R	IOBASE	CLOCK	DEVALC	IOSPGD
			QIO		
IOC$GL_DPTLIST	0002B710-R	IOBASE	QIO		
IOC$GL_IRPFL	00019F28-R	MEMALC	QIOREQ		
IOC$GL_PSBL	0001081C-R	ENTRY	CANCEL	COMDRVSUB	IOPOST
			IOSNPG	QIOREQ	
IOC$GL_PSFL	00010818-R	ENTRY	IOPOST		
IOC$INITBUFWIND	0002AD2A-R	BUFCTL	ENTRY		
IOC$INITDRV	0002C3A8-R	RELOCDRV	QIO		
IOC$INITIATE	0002B835-R	IOSNPG	ENTRY	QIOREQ	
IOC$IOPOST	00023654-R	IOPOST	QIO	SCB	
IOC$K_DIAGPID	00660000	KERNEL	CLOCK		
IOC$K_IOSPACE	40000000	ONLY730	CONFIG	DISPAT	
IOC$LOADMBAMAP	0002BD24-R	LODMAP	ENTRY		
IOC$LOADUBAMAP	0002BD85-R	LODMAP	ENTRY		
IOC$LOADUBAMAPA	0002BD6E-R	LODMAP	ENTRY		
IOC$MAPVBLK	0002BC47-R	IOSRAM	IOPOST		
IOC$MOVFRUSER	0002AD31-R	BUFCTL	ENTRY		
IOC$MOVFRUSER1	0002AD42-R	BUFCTL	ENTRY		
IOC$MOVFRUSER2	0002AD35-R	BUFCTL	ENTRY		
IOC$MOVTOUSER	0002AD49-R	BUFCTL	ENTRY		
IOC$MOVTOUSER1	0002AD5A-R	BUFCTL	ENTRY		
IOC$MOVTOUSER2	0002AD4D-R	BUFCTL	ENTRY		
IOC$PTETOPFN	0002BE01-R	LODMAP	ENTRY	IOSRAM	
IOC$PURGDATAP	00024EAA-R	ONLY730	ENTRY		
IOC$PUTBYTE	0002AD19-R	BUFCTL	ENTRY		
IOC$QNXTSEG	0002379C-R	IOPOST			
IOC$QNXTSEG1	000237A8-R	IOPOST			
IOC$REINITDRV	0002C3AE-R	RELOCDRV	QIO		
IOC$RELCHAN	0002B76A-R	IOSNPG	ENTRY		
IOC$RELDATAP	0002B868-R	IOSNPG	ENTRY		
IOC$RELMAPREG	0002B8F5-R	IOSNPG	ENTRY		
IOC$RELSCHAN	0002B760-R	IOSNPG	ENTRY		
IOC$REQCOM	0002B7F5-R	IOSNPG	ENTRY		
IOC$REQDATAP	0002B8BC-R	IOSNPG	ENTRY		
IOC$REQDATAPNW	0002B8B8-R	IOSNPG	ENTRY		
IOC$REQMAPREG	0002B92F-R	IOSNPG	ENTRY		
IOC$REQPCHANH	0002B7C1-R	IOSNPG	ENTRY		
IOC$REQPCHANL	0002B7CA-R	IOSNPG	ENTRY		
IOC$REQSCHANH	0002B7AD-R	IOSNPG	ENTRY		
IOC$REQSCHANL	0002B7B7-R	IOSNPG	ENTRY		
IOC$RETURN	0002B9C5-R	IOSNPG	ENTRY		
IOC$RTN	00010930-R	ENTRY	QIO		
IOC$SEARCHDEV	0002BA7C-R	IOSPGD	ASSIGN	DEVALC	
IOC$SEARCHGEN	0002BA81-R	IOSPGD	DEVALC		
IOC$SENSEDISK	0002BD16-R	IOSRAM	ENTRY		
IOC$UNLOCK	0002BB73-R	IOSPGD	ASSIGN	DASSGN	DEVALC

Symbol	Value	Defined By	Referenced By ...		
IOC$UPDATRANSP	0002BCDA-R	IOSRAM	ENTRY		
IOC$VERIFYCHAN	0002BB77-R	IOSPGD	CANCEL	DASSGN	GTCHAN
			QIOREQ		
IOC$WAKACP	00023806-R	IOPOST			
IOC$WFIKPCH	0002B9C6-R	IOSNPG	ENTRY		
IOC$WFIRLCH	0002B9E8-R	IOSNPG	ENTRY		
IOGEN$CNTRL_INI	000274E6-R	QIO			
IOGEN$CONN_VEC	00024F0B-R	ONLY730	QIO		
ISTKPTR	0002F800-R	KERNEL	EXCEPT	MEMMGT	SHOWMEMORY
KB_CHECK	0001AB36-R	KERNEL	ANSI	APT	ATTACH
			BUFFER	CLI	CLOCK
			DIRECTORY	DISPAT	ENTRY
			EVENT	LOOP	PRINT
			SCB		
KB_CHECK_APT	0001AB3C-R	KERNEL	CLI		
KB_POLL	0001E86C-R	CONSOLE	CSDDBTDRV	KERNEL	
KSTKPTR	0002F000-R	KERNEL	MEMMGT	SHOWMEMORY	
LIB$CVT_DTB	000293DD-R	LIB$CVT_ATB	CVTFILNAM	FILEREAD	
LIB$CVT_HTB	000293ED-R	LIB$CVT_ATB			
LIB$CVT_OTB	000293E4-R	LIB$CVT_ATB			
LINE_COUNT	00019F65-R	PRINT	CLI		
LOC$ADAPTER	0001C9D2-R	ATTACH			
LOC$PTABLE	0001CA27-R	ATTACH	DIRECTORY	RMS	
LOC$PTDESC	0001CA76-R	ATTACH	QIO		
MAP$IOSPACE	00024E28-R	ONLY730	MEMMGT		
MAPFREE	00024426-R	MEMMGT	CLI	CRDIMAGE	LOAD
			START		
MAPMEM	0002417C-R	MEMMGT	KERNEL	SHOWMEMORY	
MAPMEM$IOSPACE	0002432B-R	MEMMGT	ONLY730		
MAP_DEBUGGER	000248C9-R	MEMMGT	DEBUG	START	
MBA$INITIAL	0002BECF-R	MBAINT	QIO		
MBA$INT	0002BE24-R	MBAINT	QIO		
MEMPOOL$INIT	000240A7-R	MEMALC	KERNEL	MEMMGT	
MMG$GL_POBASE	00010808-R	ENTRY			
MMG$GL_SPTBASE	00010804-R	ENTRY	IOSNPG	IOSRAM	MEMMGT
MMG$IOLOCK	00024843-R	MEMMGT	QIOFDT		
MMG$PAGEFAULT	00021DEC-R	EXCEPT			
MMG$UNLOCK	0002484D-R	MEMMGT	IOPOST		
MODELST	000131FD-R	DEBUG	EXCEPT	MCHK730	
MS$CMD	00018540-R	TSBTDRIVR			
MT$CHECK_ACCESS	000293B5-R	VMSDUMMY	ENTRY		
ODS$GET	00024B70-R	ODS			
ODS$OPEN	00024A7F-R	ODS	RMS		
ODS$READ	00024C99-R	ODS			
ONLY_CLOCK_INIT	00024FCB-R	ONLY730	CLOCK		
ONLY_SCB	00024F44-R	ONLY730	SCB		
POPT_BASE	00030A00-R	KERNEL	ENTRY	MEMMGT	SHOWMEMORY
PARSE_DEVICE	0001BF46-R	ATTACH			
PCB_BASE	0002C678-R	KERNEL	EXCEPT	SHOWMEMORY	
PFN$AB_STATE	00019F40-R	MEMMGT			
PFN$AB_TYPE	00000000	MEMMGT			
PFN$AL_BAK	00019F38-R	MEMMGT			

Symbol	Value	Defined By	Referenced By ...		
PFN$AL_PTE	00019F34-R	MEMMGT			
PFN$AW_BLINK	00000000	MEMMGT			
PFN$AW_FLINK	00000000	MEMMGT			
PFN$AW_REFCNT	00019F3C-R	MEMMGT			
PFN$AW_SWPVBN	00019F30-R	MEMMGT			
PHD_BASE	0002C600-R	KERNEL	SHOWMEMORY		
PR$_MAPEN	00000038	SYS$PRDEF	APT		
PREADVBLK	0001EAA4-R	CONSOLE	QIO		
PRT$C_UR	0000000F	SYS$PRDEF	DEBUG		
PRT$C_UW	00000004	SYS$PRDEF	DEBUG		
PT$EXTRACT	0001C34C-R	ATTACH			
PT$INTERPRET	0001C07E-R	ATTACH			
QA$ADD_FORCED_ERROR	000266DE-R	QAMAIN	QACHECKS		
QA$ADD_QA_ERROR	000266E1-R	QAMAIN	QACHECKS		
QA$AOB_CHECK_STATE	0001A020-R	QAMAIN	DISPAT START	LOOP	QACHECKS
QA$AOB_FLAGS	0001A024-R	QAMAIN	DISPAT START	PARAM	QACHECKS
QA$AOB_ROUTINE_STATE	0001A01C-R	QAMAIN	QACHECKS		
QA$FIND_USER_CALL_FRAME	0002687F-R	QAMAIN	QACHECKS		
QA$LOOP_ON_SUBTEST	0002622C-R	QACHECKS	QAMAIN		
QA$LOOP_ON_TEST	0002612A-R	QACHECKS	QAMAIN		
QA$L_SAVED_DSA_FLAGS	0001A018-R	QAMAIN			
QA$MAIN	000268FF-R	QAMAIN	DISPAT LOOP	ERROR PARAM	KERNEL START
QA$MULTIPLE_PASS	0002605B-R	QACHECKS	QAMAIN		
QA$NORMAL_START	00025F90-R	QACHECKS	QAMAIN		
QA$PRINT_FORCED_ERRORS	00026AD8-R	QAMAIN	QACHECKS		
QA$RUN_BACKWARDS	0002634C-R	QACHECKS	QAMAIN		
QA$T_HEADER_1	0001744F-R	QACHECKS	QAMAIN		
QA$T_HEADER_2	00017475-R	QACHECKS	QAMAIN		
QA$T_HEADER_3	0001749C-R	QACHECKS	QAMAIN		
QA$T_OPERATOR_REQUEST_MESSAGE	000172C9-R	QACHECKS	QAMAIN		
QA$W_BRANCH	0001A00E-R	QAMAIN	QACHECKS		
QA$W_ERROR_NUMBER	0001A010-R	QAMAIN	QACHECKS		
QA$W_MULTIPLEPASS	00019FEC-R	QAFLAGS	QACHECKS	QAMAIN	
QA$W_SAVE_LAST	0001A014-R	QAMAIN	DISPAT	QACHECKS	
QA$W_SAVE_TEST	0001A012-R	QAMAIN	QACHECKS		
QA$W_SUBTESTLOOPS	00019FEA-R	QAFLAGS	QACHECKS	QAMAIN	
QA$W_TESTLOOPS	00019FE8-R	QAFLAGS	QACHECKS	QAMAIN	
QIO$ADDUNIT	00026CFA-R	QIO	START		
QIO$CLEANUP	00026C33-R	QIO	DIRECTORY START	DISPAT	SCRIPT
QIO$DEALLOCATE	00027574-R	QIO	ONLY730		
QIO$INITIALIZE	00026BDF-R	QIO	KERNEL		
QIO$MINCORE	0000SE3C	KERNEL	MEMALC		
QIOCLEAN_CPU	00024F16-R	ONLY730	QIO		
Q_ADAPTER	000190EE-R	ATTACH	CLI		
RCTRLC	0001AA97-R	KERNEL	PRINT		
READVBLK	0001EAA4-R	CONSOLE	QIO		
RMS$ALLOC_CACHE	000278D8-R	RMS	ANSI	ODS	RT11
RMS$CLEANUP	00027937-R	RMS	DISPAT		

Symbol	Value	Defined By	Referenced By ...		
RMS$CLOSE	00027E4B-R	RMS	ENTRY		
RMS$CONNECT	00027C43-R	RMS	ENTRY		
RMS$DISCONNECT	00027DF1-R	RMS	ENTRY		
RMS$ERROR	000277F7-R	RMS			
RMS$FILE_TABLE	0001A038-R	RMS			
RMS$GET	00027CC5-R	RMS	ENTRY		
RMS$INIT	00027928-R	RMS	KERNEL		
RMS$LOAD_XAB	00027889-R	RMS			
RMS$OPEN	000279BB-R	RMS	ENTRY		
RMS$READ	00027D62-R	RMS	ENTRY		
RPTEADR	00024801-R	MEMMGT			
RT11$GET	00028067-R	RT11			
RT11$OPEN	00027F71-R	RT11	RMS		
RT11$READ	00028177-R	RT11			
SCAN$ALPHA	0002569E-R	PARSE	ATTACH	QIO	
SCAN$ALPHANUM	000256B0-R	PARSE	ATTACH		
SCAN$BINARY	00025750-R	PARSE			
SCAN$COMPARE	00025703-R	PARSE	SCRIPT		
SCAN$DECIMAL	00025746-R	PARSE	ATTACH		
SCAN$DEVICE	00025843-R	PARSE	ATTACH	DIRECTORY	LOAD
			RMS		
SCAN$FILE	000256C2-R	PARSE			
SCAN$HEX	00025755-R	PARSE			
SCAN$NUMERIC	00025758-R	PARSE	ANSI	ATTACH	DIRECTORY
			DSVECS	PARAM	
SCAN$OCTAL	0002574B-R	PARSE			
SCAN$SEARCHLIST	000257F4-R	PARSE	QIO		
SCAN$SPACE	000256D4-R	PARSE			
SCAN$SPACES	000256D4-R	PARSE	ATTACH	PARAM	SCRIPT
SCAN$SPANC	000256E8-R	PARSE			
SCAN$SYMBOL	0002568C-R	PARSE	ATTACH		
SCBVECTOR_000	00024F49-R	ONLY730	SCB		
SCBVECTOR_038	00024F4D-R	ONLY730	SCB		
SCBVECTOR_03C	00024F51-R	ONLY730	SCB		
SCBVECTOR_050	00024F55-R	ONLY730	SCB		
SCBVECTOR_054	00024F59-R	ONLY730	SCB		
SCBVECTOR_058	00024F5D-R	ONLY730	SCB		
SCBVECTOR_05C	00024F61-R	ONLY730	SCB		
SCBVECTOR_060	00024F65-R	ONLY730	SCB		
SCBVECTOR_064	00024F69-R	ONLY730	SCB		
SCBVECTOR_068	00024F6D-R	ONLY730	SCB		
SCBVECTOR_06C	00024F71-R	ONLY730	SCB		
SCBVECTOR_070	00024F75-R	ONLY730	SCB		
SCBVECTOR_074	00024F79-R	ONLY730	SCB		
SCBVECTOR_078	00024F7D-R	ONLY730	SCB		
SCBVECTOR_07C	00024F81-R	ONLY730	SCB		
SCBVECTOR_080	00024F85-R	ONLY730	SCB		
SCBVECTOR_0C4	00024F89-R	ONLY730	SCB		
SCBVECTOR_0C8	00024F8D-R	ONLY730	SCB		
SCBVECTOR_0CC	00024F91-R	ONLY730	SCB		
SCBVECTOR_0D0	00024F95-R	ONLY730	SCB		
SCBVECTOR_0D4	00024F99-R	ONLY730	SCB		

Symbol	Value	Defined By	Referenced By ...		
SCBVECTOR_0D8	00024F9D-R	ONLY730	SCB		
SCBVECTOR_0DC	00024FA1-R	ONLY730	SCB		
SCBVECTOR_0E0	00024FA5-R	ONLY730	SCB		
SCBVECTOR_0E4	00024FA9-R	ONLY730	SCB		
SCBVECTOR_0E8	00024FAD-R	ONLY730	SCB		
SCBVECTOR_0EC	00024FB1-R	ONLY730	SCB		
SCBVECTOR_0F0	00024FB5-R	ONLY730	SCB		
SCBVECTOR_0F4	00024FB9-R	ONLY730	SCB		
SCBVECTOR_0F8	00024FBD-R	ONLY730	SCB		
SCBVECTOR_0FC	00024FC1-R	ONLY730	SCB		
SCB_BASE	0002CA00-R	KERNEL	CHAN730	CONFIG	DISPAT
			ONLY730	QIO	SCB
			SHOWMEMORY		
SCB_IMAGE	00029600-R	SCB	CONFIG	EXCEPT	KERNEL
			ONLY730	QIO	
SCB_UNKINT	0002D200-R	KERNEL	CHAN730	MEMMGT	ONLY730
			SCB		
SCH$ASTDEL	0001B8CC-R	ASTDEL	SCB		
SCH$GL_CURPCB	00018DA0-R	KERNEL	ASTDEL		
SCH$GL_PCBVEC	0001080C-R	ENTRY	ASSIGN	ASTDEL	IOPOST
SCH$LOCKW	000293BC-R	VMSDUMMY	ENTRY		
SCH$NEWLVL	0001B8F9-R	ASTDEL	ASTCON	CHMK	
SCH$POSTEF	000218C3-R	EVENT	ENTRY	IOPOST	
SCH$QAST	0001B9ED-R	ASTDEL	CLOCK	COMDRVSUB	ENTRY
			IOPOST		
SCH$UNLOCK	000293C3-R	VMSDUMMY	ENTRY		
SCH$WAKE	000293B8-R	VMSDUMMY			
SCRIPT$CONT	00028645-R	SCRIPT	CLI		
SCRIPT$FLUSH	00028621-R	SCRIPT	ATTACH	CLI	DIRECTORY
			LOAD	SHOWMEMORY	START
SCRIPT$INIT	000284D0-R	SCRIPT	KERNEL		
SCRIPT$MINCORE	00000400	SCRIPT	MEMALC		
SCRIPT$OPEN	000284DA-R	SCRIPT	CLI		
SCRIPT$STOP	0002863C-R	SCRIPT	DEBUG	EXCEPT	KERNEL
SEC_TICK	00000064	CLOCK			
SET_ABORT	00018B69-R	ATTACH			
SGN$C_IRPCNT	00000040	KERNEL	MEMALC		
SGN$C_NPAGEDYN	0000263C	KERNEL			
SHOWMEMORYFLAGS	0001A168-R	SHOWMEMORY	CLI		
SIZE_TERM	0001EC51-R	CONSOLE	KERNEL		
SS$_ACCVIO	0000000C	SYS$SSDEF	DEBUG	ERROR	IOPOST
			QIOREQ		
SS$_BADPARAM	00000014	SYS$SSDEF	ACPFDT		
SS$_BREAK	00000414	SYS$SSDEF	DEBUG		
SS$_CANCEL	00000830	SYS$SSDEF	CANCEL		
SS$_CONTINUE	00000001	SYS$SSDEF	DEBUG	ONLY730	
SS$_CONTROLC	00000651	SYS$SSDEF	CONSOLE	ONLY730	SCRIPT
SS$_CTRLERR	00000054	SYS$SSDEF	ERROR	ONLY730	
SS$_DATACHECK	0000005C	SYS$SSDEF	ONLY730		
SS$_DEVOFFLINE	00000084	SYS$SSDEF	QIOREQ		
SS$_ENDOFFILE	00000870	SYS$SSDEF	ACPFDT	CLI	SCRIPT
SS$_FILNOTACC	000000AC	SYS$SSDEF	ACPFDT		

Symbol	Value	Defined By	Referenced By ...		
SS$_ILLBLKNUM	000000DC	SYS$SSDEF	ACPFDT	IOPOST	
SS$_ILLIOFUNC	000000F4	SYS$SSDEF	QIOREQ	SCRIPT	
SS$_INSFMEM	00000124	SYS$SSDEF	CANCEL	QIO	
SS$_IVCHAN	0000013C	SYS$SSDEF	QIO		
SS$_NOPRIV	00000024	SYS$SSDEF	ACPFDT		
SS$_NORMAL	00000001	SYS$SSDEF	ACPFDT	BUFFER	CANCEL
			COMDRVSUB	IOSNPG	IOSRAM
			QIO	QIOREQ	SCB
			SCRIPT		
SS$_NOSUCHDEV	00000908	SYS$SSDEF	ERROR		
SS$_NOSUCHFILE	00000910	SYS$SSDEF	ERROR		
SS$_NOTRAN	00000629	SYS$SSDEF	KERNEL		
SS$_PAGOWNVIO	000001EC	SYS$SSDEF	BUFFER		
SS$_RESIGNAL	00000918	SYS$SSDEF	DEBUG	ONLY730	
SS$_ROPRAND	00000454	SYS$SSDEF	DEBUG		
SS$_TBIT	00000464	SYS$SSDEF	DEBUG		
SS$_WASSET	00000009	SYS$SSDEF	CONSOLE		
SSTKPTR	00030400-R	KERNEL	EXCEPT	MEMMGT	SHOWMEMORY
STACK_BASE	0002DA00-R	KERNEL	EXCEPT	MEMMGT	SHOWMEMORY
SWI$GL_FQBL	00018DA8-R	KERNEL			
SWI$GL_FQFL	00018DA4-R	KERNEL	FRKCTL		
SYMTAB_HI	00019949-R	DEBUG	MEMMGT		
SYS$ALLOC	00010238-R	ENTRY	ATTACH		
SYS$ASCTIM	00010248-R	ENTRY	FAO		
SYS$ASSIGN	00010250-R	ENTRY	KERNEL		
SYS$BINTIM	00010258-R	ENTRY	CONSOLE	KERNEL	
SYS$CALL_HANDL	00010210-R	ENTRY	EXCEPT	UNWIND	
SYS$CANCEL	00010260-R	ENTRY	DASSGN	PRINT	
SYS$CANTIM	00010268-R	ENTRY	CLOCK	DISPAT	
SYS$CLOSE	000105B8-R	ENTRY	DIRECTORY	DSLOAD	HELP
			SCRIPT		
SYS$CLREF	00010298-R	ENTRY	CLOCK	FLAGS	KERNEL
			LOAD	QIOREQ	
SYS$CNTREG	000102A0-R	ENTRY	BUFFER	CRDIMAGE	
SYS$CONNECT	000105C0-R	ENTRY	DIRECTORY	DSLOAD	HELP
			KERNEL	SCRIPT	
SYS$CREATE	7FFEE1C8	SYS$P1_VECTOR	KERNEL		
SYS$CRELOG	7FFEDEB0	SYS$P1_VECTOR	KERNEL	LOAD	
SYS$CRETVA	800000C8	KERNEL	DEBUG		
SYS$DALLOC	000102D8-R	ENTRY	ATTACH	KERNEL	
SYS$DASSGN	000102E0-R	ENTRY			
SYS$DCLEXH	7FFEDEF0	SYS$P1_VECTOR	KERNEL		
SYS$DELTVA	80000110	KERNEL			
SYS$DISCONNECT	000105D0-R	ENTRY	DSLOAD	SCRIPT	
SYS$DISK	00019E5C-R	LOAD	KERNEL		
SYS$DISMOU	00000000-*		DEVALC		
SYS$EXIT	00010340-R	ENTRY	CLI	KERNEL	
SYS$EXPREG	00010348-R	ENTRY	BUFFER	CRDIMAGE	MEMALC
SYS$FAO	00010350-R	ENTRY	ANSI	ATTACH	CVRTIM
			DEBUG	DIRECTORY	ERROR
			IOSPGD	PARAM	PRINT
			SCRIPT		

Symbol	Value	Defined By	Referenced By ...		
SYS$FAOL	00010358-R	ENTRY	ATTACH	DEBUG	PRINT
SYS$GET	00010580-R	ENTRY	DIRECTORY	HELP	SCRIPT
SYS$GETCHN	000104C8-R	ENTRY	KERNEL		
SYS$GETSYI	7FFEE3F8	SYS$P1_VECTOR	KERNEL		
SYS$GETTIM	00010378-R	ENTRY	CLOCK		
SYS$GL_OPRMBX	00010800-R	ENTRY			
SYS$GTCHAN	00010380-R	ENTRY			
SYS$HIBER	00010388-R	ENTRY	CLOCK	SCRIPT	
SYS$LKWSET	000103A0-R	ENTRY			
SYS$MOUNT	00000000-*		DEVALC		
SYS$NUMTIM	000103B8-R	ENTRY	CVRTIM		
SYS$OPEN	00010608-R	ENTRY	DIRECTORY	DSLOAD	HELP
			KERNEL	SCRIPT	
SYS$PARSE	7FFEE230	SYS$P1_VECTOR	DIRECTORY		
SYS$PUT	7FFEE188	SYS$P1_VECTOR	PRINT		
SYS$QIO	000103C8-R	ENTRY	EXCEPT	SCRIPT	
SYS$QIOW	00010200-R	ENTRY	CONSOLE	ERROR	KERNEL
			PRINT	SCRIPT	
SYS$READ	00010590-R	ENTRY	DSLOAD		
SYS$READEF	000103D0-R	ENTRY	CONSOLE	FLAGS	SCRIPT
SYS$SEARCH	7FFEE248	SYS$P1_VECTOR	DIRECTORY		
SYS$SETAST	000103F8-R	ENTRY	KERNEL		
SYS$SETDDIR	7FFEE250	SYS$P1_VECTOR	KERNEL	LOAD	
SYS$SETEF	00010400-R	ENTRY	CLOCK	CONSOLE	FLAGS
			QIOREQ		
SYS$SETEXV	80000208	KERNEL	EXCEPT		
SYS$SETIMR	00010420-R	ENTRY	CLOCK	CONSOLE	
SYS$SETPRI	00010428-R	ENTRY			
SYS$SETPRT	00010430-R	ENTRY	DEBUG	ONLY730	
SYS$SETRWM	00010438-R	ENTRY			
SYS$SPACE	7FFEE218	SYS$P1_VECTOR	DSLOAD		
SYS$SYSROOT	00019E49-R	LOAD	KERNEL		
SYS$TRNLOG	00010458-R	ENTRY	KERNEL		
SYS$ULKPAG	00010460-R	ENTRY			
SYS$ULWSET	00010468-R	ENTRY			
SYS$UNWIND	00010470-R	ENTRY	CONFIG	DEBUG	EXCEPT
			MEMMGT	ONLY730	PROBE
SYS$WAITFR	00010478-R	ENTRY			
SYS$WAKE	00010480-R	ENTRY	CLOCK	SCRIPT	
SYS$WFLAND	00010488-R	ENTRY			
SYS$WFLOR	00010490-R	ENTRY			
TS11_DRIVER	0002A2E8-R	TSBTDRIVR			
TST$MCHK	00021D74-R	EXCEPT	DISPAT	ONLY730	
TU_INIT	0002A600-R	MUBTDRIVR			
T_EMESSG1	000111D4-R	CLI	DEBUG		
UBA$INITIAL	00024ECA-R	ONLY730	QIO		
UBA$UNEXINT	00024ECC-R	ONLY730	QIO		
UBADP_CPU	00024EBD-R	ONLY730	QIO		
UBINTSZ	000000B4	ONLY730	QIO		
UBINT_DISP	00017BD0-R	QIO	ONLY730		
UDA_RESPONSE	0002A250-R	PUBTDRIVR	CONFIG		
UD_INIT	0002A000-R	PUBTDRIVR	CONFIG		

Symbol	Value	Defined By	Referenced By ...		
UNMAP_DEBUGGER	000248E2-R	MEMMGT	DEBUG		
USTKPTR	00030A00-R	KERNEL	EXCEPT	MEMMGT	SHOWMEMORY
VMS$QIO	0002C177-R	QIOREQ	QIO		
VRABORT	0002924E-R	START	APT	CLI	
VRCLRBRK	0001FBD9-R	DEBUG	APT	CLI	
VRCLREF	000223DC-R	FLAGS	CLI		
VRCLRFLG	000223A7-R	FLAGS	CLI		
VRDEPOSIT	0001FA88-R	DEBUG	CLI		
VREXAMINE	0001F7A0-R	DEBUG	CLI		
VRRESTART	00028ED0-R	START	APT	DISPAT	LOOP
VRSETBASE	0001F6D6-R	DEBUG	CLI		
VRSETBRK	0001FB83-R	DEBUG	CLI		
VRSETDFLT	0001F6F3-R	DEBUG	CLI		
VRSETEF	000223BB-R	FLAGS	CLI		
VRSETFLG	00022384-R	FLAGS	CLI		
VRSETMEM	00028E21-R	SHOWMEMORY	CLI		
VRSETQA	00026428-R	QAFLAGS	CLI		
VRSETWIDTH	0001E7C0-R	CONSOLE	CLI	KERNEL	
VRSHOWBASE	0001F6DF-R	DEBUG	CLI		
VRSHOWBRK	0001FC52-R	DEBUG	CLI		
VRSHOWDFLT	0001F736-R	DEBUG	CLI		
VRSHOWEF	00022491-R	FLAGS	CLI	QAMAIN	
VRSHOWFLG	000223FD-R	FLAGS	CLI	QAMAIN	
VRSHOWQA	000264D5-R	QAFLAGS	CLI		
VRSHOWSECTIONS	0002935B-R	START	CLI		
VRSHOWSTATUS	000210CF-R	DISPAT	CLI		
VRSHOWWIDTH	0001E84C-R	CONSOLE	CLI		
VRSHOW_CALLS	0001CC7C-R	CALLFRAME	CLI		
VRSTART	00028E7F-R	START	APT	CLI	LOAD
VRSUMMARY	00021057-R	DISPAT	CLI		
VRSUPPORT	00029301-R	START	CLI		
WIDE_FLAG	00019959-R	DIRECTORY	CLI		
WRITEVBLK	0001EA88-R	CONSOLE	QIO		
XDELBPT	00000000		WK-CLI	WK-KERNEL	
XDELTBIT	000C0000		WK-KERNEL		
XDS$INIT	00000000	MEMMGT			

Symbols By Value

Value	Symbols...		
00000000	CRD$K_CRD_INITIALIZATION	CRD$K_DS_FLAGS	CRD$K_HOOK_POINT
	CRD$K_READ	DS$K_BLSS_B	PFN$AB_TYPE
	PFN$AW_BLINK	PFN$AW_FLINK	XDS$INIT
00000001	CRD$K_DS_CONTROL_C	CRD$K_LAST_HOOK_POINT	CRD$K_TYPECODE
	CRD$K_WRITE	DS$K_BLSSU_B	DYN$C_ADP
	SS$_CONTINUE	SS$_NORMAL	
00000002	CRD$K_LAST_ACCESS_CODE	CRD$K_LAST_DS_VARIABLE	CRD$K_LAST_FUNCTION
	DS$K_BLEQ_B	DYN$C_ACB	
00000003	BPTCODE	DS$K_BLEQU_B	DYN$C_AQB
00000004	DS$K_BEQL_B	DYN$C_CEB	PRT$C_UW
00000005	DEV$V_SQD	DS$K_BEQLU_B	DYN$C_CRB
00000006	DEV$V_SPL	DS$K_BGEQ_B	DYN$C_DDB
	IO$V_CANCTRLO		
00000007	DS$K_BGEQU_B	DYN$C_FCB	
00000008	DS$K_BGTR_B	DYN$C_FRK	
00000009	DS$K_BGTRU_B	DYN$C_IDB	SS$_WASSET
0000000A	DS$K_BNEQ_B	DYN$C_IRP	
0000000B	DS$K_BNEQU_B	DYN$C_LOG	
0000000C	DS$K_BLSS_M	DYN$C_PCB	SS$_ACCVIO
0000000D	DEV$V_NET	DS$K_BLSSU_M	DYN$C_PQB
0000000E	DEV$V_FOD	DS$K_BLEQ_M	DYN$C_RVT
0000000F	BPTMAX	DS$K_ASCIBUF	DS$K_BLEQU_M
	DYN$C_TQE	PRT$C_UR	
00000010	DEV$V_SHR	DS$K_BEQL_M	DYN$C_UCB
00000011	DFF$C_BKSTP_LEN	DS$K_BEQLU_M	DYN$C_VCB
00000012	DS$K_BGEQ_M	DYN$C_WCB	
00000013	DEV$V_MNT	DS$K_BGEQU_M	DYN$C_BUFIO
00000014	DEV$V_MBX	DS$K_BGTR_M	DS$K_NYSIZ
	DYN$C_TYPAHD	HP$C_NAMELEN	SS$_BADPARAM
00000015	DEV$V_DMT	DS$K_BGTRU_M	DYN$C_GSD
00000016	DS$K_BNEQ_M	DYN$C_MVL	
00000017	DEV$V_ALL	DS$GK_SUPPORT_TABLE_ROWS	DS$K_BNEQU_M
	DYN$C_NET		
00000018	DEV$V_FOR	DS$K_BVS_M	DYN$C_KFI
00000019	DS$K_BVC_M	DYN$C_MTL	
0000001A	DS$K_BCS_M	DYN$C_BRDCST	
0000001B	DS$K_BCC_M	DYN$C_CXB	
0000001C	DS$K_BBS	DYN$C_NDB	
0000001D	DS$K_BBC	DYN$C_SSB	
0000001E	DS$K_BBSS	DYN$C_DPT	
0000001F	DS$K_BBCS	DYN$C_JPB	IO$_PHYSICAL
00000020	DS$K_BBSC	DYN$C_PBH	
00000021	DS$K_BBCC	DYN$C_PDB	IO$_READLBLK
00000022	DS$K_BBSSI	DYN$C_PIB	
00000023	DS$K_BBCCI	DYN$C_PFL	IO$_SETMODE
00000024	DS$K_BLBS	DYN$C_SFT	FIL$C_DIR_SIZE
	SS$_NOPRIV		
00000025	DS$K_BLBC	DYN$C_PTR	
00000026	DS$K_BERROR	DYN$C_KFH	

Value Symbols...

00000027	DS$K_BNERROR	DYN$C_RVX	IO$_SENSEMODE
00000028	DYN$C_EXTGSD		
00000029	DYN$C_SHMGSD		
0000002A	DYN$C_SHB		
0000002B	DYN$C_MBX		
0000002C	DYN$C_IRPE		
0000002D	DYN$C_SLAVCEB		
0000002E	DYN$C_SHMCEB		
0000002F	DYN$C_JIB	IO$_LOGICAL	
00000030	DYN$C_TWP	IO$_WRITEVBLK	
00000031	DYN$C_RBM	IO$_READVBLK	
00000032	DYN$C_VCA		
00000037	IO$_READPROMPT		
00000038	PR$_MAPEN		
0000003F	IO$M_FCODE		
00000040	DS$K_CCB	IO$M_CANCTRLO	IO$M_NOW
	SGN$C_IRPCNT		
00000054	SS$_CTRLERR		
0000005C	SS$_DATACHECK		
00000064	SEC_TICK		
00000080	DS$R_STRBUF	DYN$C_SHRBUFIO	DYN$C_SPECIAL
00000084	SS$_DEVOFFLINE		
000000AC	SS$_FILNOTACC		
000000B4	UBINTSZ		
000000DC	SS$_ILLBLKNUM		
000000F4	SS$_ILLIOFUNC		
00000100	IO$M_CTRLCAST		
00000124	SS$_INSFMEM		
0000013C	SS$_IVCHAN		
000001EC	SS$_PAGOWNVIO		
00000200	DS$A_PRGBGN	DS$GK_USOVR	DS$K_BUFSIZ
00000218	FIL$C_SIZE		
00000400	SCRIPT$MINCORE		
00000414	SS$_BREAK		
00000454	SS$_ROPRAND		
00000464	SS$_TBIT		
00000600	DS$R_SSSIZE		
00000629	SS$_NOTRAN		
00000651	SS$_CONTROLC		
00000830	SS$_CANCEL		
00000870	SS$_ENDOFFILE		
00000908	SS$_NOSUCHDEV		
00000910	SS$_NOSUCHFILE		
00000918	SS$_RESIGNAL		
00001000	IO$M_TRMNOECHO		
0000263C	SGN$C_NPAGEDYN		
00003000	DSA$M_DFLTFLGS		
00005E3C	QIO$MINCORE		
0000F800	DS$K_PRGSIZ		
00010010	R-DS$ENDPASS		
00010018	R-DS$GPHARD		
00010020	R-DS$ABORT		

Value	Symbols...		
00010028	R-DS$DOSUMMARY	R-DS$SUMMARY	
00010030	R-DS$BGNSUB		
00010038	R-DS$ENDSUB		
00010040	R-DS$CKLOOP		
00010048	R-DS$INLOOP		
00010050	R-DS$ESCAPE		
00010058	R-DS$BREAK		
00010060	R-DS$WAITMS		
00010068	R-DS$WAITUS		
00010070	R-DS$ABORTWAIT	R-DS$CANWAIT	
00010078	R-DS$CNTRLC		
00010080	R-DS$ASKDATA	R-DS$GETDATA	
00010088	R-DS$ASKVLD	R-DS$GETVIELD	
00010090	R-DS$ASKADR	R-DS$GETADDRESS	
00010098	R-DS$ASKLGCL	R-DS$GETLOGICAL	
000100A0	R-DS$ASKSTR	R-DS$GETSTRING	
000100A8	R-DS$BRANCH		
000100B0	R-DS$CVTREG		
000100B8	R-DS$PARSE		
000100C0	R-DS$ERRSYS		
000100C8	R-DS$ERRDEV		
000100D0	R-DS$ERRHARD		
000100D8	R-DS$ERRSOFT		
000100E0	R-DS$PRINTB		
000100E8	R-DS$PRINTX		
000100F0	R-DS$PRINTF		
000100F8	R-DS$PRINTS		
00010100	R-DS$PRINTSIG		
00010108	R-DS$ERRPREP		
00010110	R-DS$SETPRIEXV		
00010118	R-DS$MAPDBGBLOCK		
00010120	R-DS$GETBUF		
00010128	R-DS$RELBUF		
00010150	R-DS$MMON		
00010158	R-DS$MMOFF		
00010160	R-DS$SETVEC		
00010168	R-DS$CLRVEC		
00010170	R-DS$INITSCB		
00010178	R-DS$SETIPL		
00010180	R-DS$CHANNEL		
00010188	R-DS$SETMAP		
00010190	R-DS$SHOCHAN	R-DS$SHOWCHAN	
00010198	R-DS$LOAD		
000101A0	R-DS$PROBE		
000101A8	R-DS$ATTACH		
000101B0	R-DS$HELP		
000101B8	R-DS$FREEDBGSYM		
000101C0	R-DS$LOADPCS		
000101C8	R-DS$GETTERM		
000101D0	R-DS$SERVICE_DISPATCHER		
00010200	R-DS$AQ_SYSSRV	R-SYS$QIOW	
00010210	R-SYS$CALL_HANDL		

Value	Symbols...
00010238	R-SYS$ALLOC
00010248	R-SYS$ASCTIM
00010250	R-SYS$ASSIGN
00010258	R-SYS$BINTIM
00010260	R-SYS$CANCEL
00010268	R-SYS$CANTIM
00010298	R-SYS$CLREF
000102A0	R-SYS$CNTREG
000102D8	R-SYS$DALLOC
000102E0	R-SYS$DASSGN
00010340	R-SYS$EXIT
00010348	R-SYS$EXPREG
00010350	R-SYS$FAO
00010358	R-SYS$FAOL
00010378	R-SYS$GETTIM
00010380	R-SYS$GTCHAN
00010388	R-SYS$HIBER
000103A0	R-SYS$LKWSET
000103B8	R-SYS$NUMTIM
000103C8	R-SYS$QIO
000103D0	R-SYS$READEF
000103F8	R-SYS$SETAST
00010400	R-SYS$SETEF
00010420	R-SYS$SETIMR
00010428	R-SYS$SETPRI
00010430	R-SYS$SETPRT
00010438	R-SYS$SETRWM
00010458	R-SYS$TRNLOG
00010460	R-SYS$ULKPAG
00010468	R-SYS$ULWSET
00010470	R-SYS$UNWIND
00010478	R-SYS$WAITFR
00010480	R-SYS$WAKE
00010488	R-SYS$WFLAND
00010490	R-SYS$WFLOR
000104C8	R-SYS$GETCHN
00010580	R-SYS$GET
00010590	R-SYS$READ
000105B8	R-SYS$CLOSE
000105C0	R-SYS$CONNECT
000105D0	R-SYS$DISCONNECT
00010608	R-SYS$OPEN
000107FF	R-DS$AQ_SSEND
00010800	R-SYS$GL_OPRMBX
00010804	R-MMG$GL_SPTBASE
00010808	R-MMG$GL_POBASE
0001080C	R-SCH$GL_PCBVEC
00010810	R-EXE$GL_ABSTIM
00010818	R-IOC$GL_PSFL
0001081C	R-IOC$GL_PSBL
00010930	R-IOC$RTN
00010B00	R-EXE$GQ_SYSTIME

Value	Symbols...	
000111D4	R-T_EMESSG1	
0001121C	R-GT_MENU_ERROR	
000112A9	R-COMMAND_TREE	
000131BC	R-APSLMNE	
000131FD	R-MODELST	
000137B0	R-DS$SRVDISP_TABLE	
000141C3	R-AAT_ARITH	
0001497F	R-DS$GA_PTDESC	
00016BB3	R-FIL$GT_DDDEV	
00016BB4	R-FIL$GT_DDSTRING	
00017134	R-IO$GQ_PHYSICAL	
000172C9	R-QA$T_OPERATOR_REQUEST_MESSAGE	
0001744F	R-QA$T_HEADER_1	
00017475	R-QA$T_HEADER_2	
0001749C	R-QA$T_HEADER_3	
00017B90	R-EXE$GL_PWRDONE	
00017BC4	R-CTL$GW_NMIOCH	
00017BC6	R-CTL$GW_CHINDX	
00017BD0	R-UBINT_DISP	
00017C84	R-DEF$Q_BACKSTOP	
00017DC4	R-DS$GA_SUPPORT_TABLE	
00018540	R-MS$CMD	
00018560	R-DS$GQ_DAYTIM	
0001857F	R-DS$GT_START	
000185B4	R-DS$GT_VDS_VERSION	
000185BD	R-DS$GT_PROMPT	
00018600	R-DS$FAB_INPUT	
00018650	R-DS$RAB_INPUT	
00018694	R-DS$FAB_OUTPUT	
000186E4	R-DS$RAB_OUTPUT	
00018728	R-DS$GA_DS_CTRL_C_FIRST	
0001872C	R-DS$GA_DS_CTRL_C_SECOND	
00018730	R-DS$L_USERCNTRLC	
0001879C	R-IMAGE_HEADER	
0001899C	R-CTL$GL_CCB	R-IMAGE_HEADER_END
00018D9C	R-CTL$GL_CCBBASE	
00018DA0	R-DS$GL_PCBVEC	R-SCH$GL_CURPCB
00018DA4	R-SWI$GL_FQFL	
00018DA8	R-SWI$GL_FQBL	
00018DD4	R-BOOT$L_ADP	
00018DD8	R-BOOT$L_CSR	
00018DDC	R-BOOT$L_UNIT	
00018DE0	R-BOOT$L_R2	
00018DE4	R-BOOT$L_R5	
00018DE8	R-BOOT$L_DEVTYP	
00018DEC	R-DS$GL_EFC0	R-DS$GQ_EFL
00018DF0	R-DS$GL_EFC1	
00018DF4	R-DS$GA_LASTADR	
00018DF8	R-DS$GL_FLAGS	
00018DFC	R-DS$GW_TTIN	
00018DFE	R-DS$GW_TTOUT	
00018E00	R-DS$GL_ERRCNT	

Value	Symbols...	
00018E04	R-DS$GL_HARDERR_COUNT	
00018E08	R-DS$GL_PREPERR_COUNT	
00018E0C	R-DS$GL_SOFTERR_COUNT	
00018E10	R-DS$GL_DEVERR_COUNT	
00018E14	R-DS$GL_SYSERR_COUNT	
00018E18	R-DS$GL_ERRSUP_COUNT	
00018E1C	R-DS$GL_NUMTEST	
00018E20	R-DS$GL_FSTTEST	
00018E24	R-DS$GL_LSTTEST	
00018E28	R-DS$GL_SUBTEST	
00018E2C	R-DS$GA_CHKLPPC	
00018E30	R-DS$GA_LOOPADR	
00018E34	R-DS$GL_RUNTIM	
00018E38	R-DS$GL_TIMESEC	
00018E3C	R-DS$GL_WAITIM	
00018E40	R-DS$GL_RADIX	
00018E44	R-DS$GL_SIZE	
00018E48	R-DS$GL_BUFLEN	R-DS$GQ_BUFQWD
00018E4C	R-DS$GA_BUFPTR	
00018E50	R-DS$GB_BYTEBUF	
00018E51	R-DS$GB_TYPECODE	
00018E52	R-DS$GB_INHIBIT_NAMING	
00018E53	R-DS$GT_ASCIBUF	
00018E62	R-DS$GT_BUFFER	
00019062	R-DS$GT_STRBUF	
000190E2	R-DS$GL_TERMCHAR	
000190EE	R-Q_ADAPTER	
000190F8	R-DS$GQ_PHYADR	
00019100	R-DS$GL_BUFCNT	
00019118	R-CDB$BLOCK	
00019168	R-CVT$T_MBASR	
0001916C	R-CVT$T_MBAMAP	
00019170	R-CVT$T_UBADPR	
00019174	R-CVT$T_UBAMAP	
00019210	R-DS$GL_CLIBASE	
00019668	R-DS$GQ_CURYEAR	
00019670	R-DS$GT_NEWYEAR	
000196CA	R-DS$GB_WIDTH	
00019754	R-DS$GL_CRD_FLAGS	
00019758	R-CRD$GL_CRD_COMMAND_DEBUG	
0001975C	R-DS$AL_DS_DISPATCH_VECTORS	
00019760	R-DS$GA_CRD_DS_INTERFACE	
00019778	R-DS$GA_USER_ERROR_TEXT	
0001978C	R-DS$GL_CRD_DEBUG	
000197C4	R-DEBUG$GB_FLAGS	
000197C5	R-DS$AB_BPTINST	
000197D8	R-DS$AA_BPTADDR	
00019941	R-DEBUG_LO	
00019945	R-DEBUG_HI	
00019949	R-SYMTAB_HI	
00019959	P-WIDE_FLAG	
0001996C	R-DS$GQ_TESTID	

Value	Symbols...		
000199A8	R-HELPINFO		
000199D8	R-DS$GA_SELECTS		
000199EC	R-DS$GA_SELECTED		
00019BF0	R-DS$GA_PTABLE		
00019DF4	R-DIAG_FILE_NAME		
00019E49	R-SYS$SYSROOT		
00019E5C	R-SYS$DISK		
00019E6C	R-DEF$Q_DEV		
00019E92	R-DEF$Q_DIR		
00019F20	R-EXE$GL_SPLITADR		
00019F24	R-DS$GL_LOOKASIDE		
00019F28	R-IOC$GL_IRPFL		
00019F30	R-PFN$AW_SWPVBN		
00019F34	R-PFN$AL_PTE		
00019F38	R-PFN$AL_BAK		
00019F3C	R-PFN$AW_REFCNT		
00019F40	R-DS$GL_MEMSIZE	R-PFN$AB_STATE	
00019F44	R-DS$GL_ACTUAL_MEMSIZE		
00019F48	R-DS$GL_SET_MEMSIZE		
00019F4C	R-DS$GB_MM_ENB		
00019F65	R-LINE_COUNT		
00019F69	R-DS$GL_PAGE		
00019FE1	R-DEBUG_THE_SUCKER	R-DS$GB_VDS_DEBUG	
00019FE8	R-QA$W_TESTLOOPS		
00019FEA	R-QA$W_SUBTESTLOOPS		
00019FEC	R-QA$W_MULTIPLEPASS		
0001A00E	R-QA$W_BRANCH		
0001A010	R-QA$W_ERROR_NUMBER		
0001A012	R-QA$W_SAVE_TEST		
0001A014	R-QA$W_SAVE_LAST		
0001A018	R-QA$L_SAVED_DSA_FLAGS		
0001A01C	R-QA$AOB_ROUTINE_STATE		
0001A020	R-QA$AOB_CHECK_STATE		
0001A024	R-QA$AOB_FLAGS		
0001A038	R-RMS$FILE_TABLE		
0001A048	R-DS$GB_STOPPING_THE_TIMER		
0001A049	R-DS$GA_CHMKVEC		
0001A04D	R-DS$GA_BREAKVEC		
0001A051	R-DS$GA_TBITVEC		
0001A055	R-EXE$GL_SCB		
0001A059	R-DS$GA_SOFTINT		
0001A164	R-DS$GB_QIOACT		
0001A168	R-SHOWMEMORYFLAGS		
0001A200	R-BOOT		
0001A297	R-APT		
0001A400	R-INI$BRK		
0001A402	R-DS$SOFTIPL5		
0001A40D	R-DS$USERMODE		
0001A91E	R-BEGIN	R-BEGIN_BLISS	
0001A93B	R-INIT_CONTEXT		
0001AA29	R-CMK$REFRESH		
0001AA97	R-RCTRLC		

Value	Symbols...
0001AB36	R-KB_CHECK
0001AB3C	R-KB_CHECK_APT
0001AC0C	R-DSX$CNTREC
0001AC37	R-DSV$EXIT
0001ACE0	R-DSR$CHECK_MENUTEST_OFF_SET
0001AD04	R-DSR$CHECK_AUTOTEST_OFF_SET
0001AD23	R-DSX$GETTERM
0001AD34	R-ANSI$OPEN
0001B2DC	R-ANSI$CLOSE
0001B30D	R-ANSI$CACHEVBN
0001B412	R-ANSI$READ
0001B4D3	R-ANSI$GET
0001B6A0	R-GET_BLOCK_LENGTH
0001B6D0	R-APT_COM
0001B888	R-APT_MSG
0001B891	R-APT_DONE
0001B8A0	R-EXE$SETAST
0001B8CC	R-SCH$ASTDEL
0001B8F9	R-SCH$NEWLVL
0001B9ED	R-SCH$QAST
0001BA6C	R-DSV$ATTACH
0001BB69	R-SET_ABORT
0001BB76	R-DSX$ATTACH
0001BF46	R-PARSE_DEVICE
0001C07E	R-PT$INTERPRET
0001C34C	R-PT$EXTRACT
0001C50A	R-DSV$SHOWDEVICEB
0001C511	R-DSV$SHOWDEVICE
0001C56B	R-DSV$SHOWSELECTB
0001C572	R-DSV$SHOWSELECT
0001C5A3	R-DSV$DEATTACH
0001C63C	R-DSV$SELECT
0001C7C2	R-DSV$DESELECT
0001C92E	R-INI$PTABLE
0001C9D2	R-LOC$ADAPTER
0001CA27	R-LOC$PTABLE
0001CA76	R-LOC$PTDESC
0001CAD0	R-INS$PTABLE
0001CB68	R-FIL$RDWRTLBN
0001CB88	R-BOO$IMAGE_ATT
0001CBBC	R-DSX$GETBUF
0001CC0E	R-DSX$RELBUF
0001CC7C	R-VRSHOW_CALLS
0001CD3B	R-DSR$FIND_USER_PC
0001CD6C	R-DSX$CHANNEL
0001CE1D	R-DSX$SETMAP
0001CFED	R-DSP$SHOWMBA
0001CFF7	R-DSP$SHOWUBI
0001D008	R-DSX$SHOWCHAN
0001D0EC	R-DSI$CHANNEL_VEC
0001D154	R-EXE_CMODKRNL
0001D1A8	R-CMK$ASTEXIT

Value	Symbols...	
0001D1DD	R-DS$CLI	
0001D31A	R-CLI_ACTION	
0001D4C6	R-ACT_CONT	
0001D4D9	R-ACT_DEFAULT_DBG	
0001DC0C	R-DSX$CANWAIT	
0001DC27	R-EXE$WAKE	
0001DC33	R-DSX$WAITMS	
0001DCBC	R-DSX$WAITUS	
0001DCD7	R-DSX$WAITUS_USOVR	
0001DD3D	R-EXE$HIBER	
0001DD5D	R-CMK$HIBER	
0001DD73	R-EXE$CANTIM	
0001DD88	R-CMK$CANTIM	
0001DDC0	R-DSR$SETIMR_INI	
0001DDDD	R-EXE$INSTIMQ	
0001DE0F	R-EXE$SETIMR	
0001DE21	R-CMK$SETIMR	
0001DE94	R-DSI$TIMSRV	
0001DEF4	R-DSI$TIMER	
0001DFE4	R-CLK$RE_INIT	
0001E034	R-INI$LOAD_DEVICE	
0001E0B3	R-DSP$CONFIG_LOAD	
0001E7C0	R-VRSETWIDTH	
0001E84C	R-VRSHOWWIDTH	
0001E86C	R-KB_POLL	
0001E8C9	R-CMK$RCONSOLE	
0001EA72	R-CMK$TCONSOLE	
0001EA88	R-WRITEVBLK	
0001EAA4	R-PREADVBLK	R-READVBLK
0001EC51	R-SIZE_TERM	
0001ED28	R-DSR$LOAD_CRD	
0001EF53	R-DSR$UNLOAD_CRD	
0001F036	R-DSR$RESTORE_CRD_VECTORS	
0001F081	R-EXCHANGE_DISPATCH_VECTORS	
0001F4AE	R-DSR$PRINT_TYPECODE_NAME	
0001F600	R-FIL$CVTFILNAM	
0001F6D6	R-VRSETBASE	
0001F6DF	R-VRSHOWBASE	
0001F6F3	R-VRSETDFLT	
0001F736	R-VRSHOWDFLT	
0001F7A0	R-VREXAMINE	
0001FA88	R-VRDEPOSIT	
0001FB83	R-VRSETBRK	
0001FBD9	R-VRCLRBRK	
0001FC29	R-DSP$REMOVEBREAK	
0001FC52	R-VRSHOWBRK	
0001FCB4	R-COND_DEBUG	
0001FE99	R-FETCH_STORE	
00020063	R-CMK$SGIPR	
00020072	R-FETCH_STOR_PREG	
000200A1	R-DSV$DEBUG	
00020209	R-DSX$GPHARD	

Value	Symbols...	
0002023B	R-DSP$GEN_PTABLES	
00020A01	R-DSX$DIRECTORY	
00020BEF	R-DSV$DIRECTORY	
00020C18	R-DISPATCH	
00020EB8	R-DSX$ENDPASS	
00020F82	R-DS_CLEANUP	
00021045	R-DSX$DOSUMMARY	
00021057	R-VRSUMMARY	
000210CF	R-VRSHOWSTATUS	
000211A0	R-DSX$LOAD	
0002139C	R-DSX$SERVICE_DISPATCHER	
000213D5	R-DSX$ERRSOFT	
000213FC	R-DSX$ERRHARD	
00021423	R-DSX$ERRPREP	
0002144A	R-DSX$ERRDEV	
00021478	R-DSX$ERRSYS	
00021649	R-DS_TESTID	
000216B5	R-DS_ERRSUP	
000217E2	R-DSR$COMPLETION	
0002188F	R-EXE$SETEF	
000218A9	R-EXE$CLREF	
000218C3	R-SCH$POSTEF	
000218D6	R-EXE$READEF	
000218FA	R-EXE$WAITFR	
00021917	R-EXE$WFLOR	
00021938	R-EXE$WFLAND	
00021964	R-COND_HANDLR	
00021A6C	R-DSX$PRINTSIG	
00021A7E	R-DSX$PRINTSIGI	
00021D31	R-FIND_USERPC	
00021D74	R-TST$MCHK	
00021D8C	R-EXE_MCHK	
00021DAC	R-EXE_KRNLSTK	
00021DB8	R-EXE_POWER	
00021DC4	R-EXE_OPCDEC	
00021DCC	R-EXE_OPCCUS	
00021DD4	R-EXE$ROPRAND	R-EXE_ROPRAND
00021DDC	R-EXE_RADRMOD	
00021DE4	R-EXE$ACVIOLAT	R-EXE_ACCVIO
00021DEC	R-EXE_TRANSL	R-MMG$PAGEFAULT
00021DF8	R-EXE_COMPAT	
00021E00	R-EXE_ARITH	
00021E0C	R-EXE$BREAK	R-EXE_BREAK
00021E38	R-EXE$TBIT	R-EXE_TBIT
00021E59	R-EXE_UNEXPINT	
00021E64	R-EXE_CHMK	
00021E6C	R-EXE_CHME	
00021E74	R-EXE_CHMS	
00021E7C	R-EXE_CHMU	
00022021	R-DSX$SETPRIEXV	
00022060	R-EXE$FAO	
00022068	R-EXE$FAOL	

Value	Symbols...
00022384	R-VRSETFLG
000223A7	R-VRCLRFLG
000223BB	R-VRSETEF
000223DC	R-VRCLREF
000223FD	R-VRSHOWFLG
00022491	R-VRSHOWEF
000224FC	R-FIL$OPENFILE
00022602	R-FIL$CACHE_INIT
00022676	R-FIL$CACHE_TRUNC
0002274D	R-FIL$MOUNT
000227DD	R-FIL$FINDFILID
00022AE4	R-FIL$RDCHKFILHDR
00022BBC	R-FIL$WRITEVBN
00022BC3	R-FIL$READVBN
00022C7A	R-FIL$STATBLK
00022D44	R-DSX$HELP
000234E9	R-DSR$KEY_EQUAL
00023654	R-IOC$IOPOST
0002379C	R-IOC$QNXTSEG
000237A8	R-IOC$QNXTSEG1
00023806	R-IOC$WAKACP
00023952	R-IOC$DIRPOST1
00023A12	R-DSV$SETLOAD
00023A8E	R-DSV$SHOWLOAD
00023AAE	R-DSR$IFLOADDEV
00023AD8	R-DSV$RUN
00023AD9	R-DSV$LOAD
00023BF9	R-DSX$ESCAPE
00023C0E	R-DSX$BGNSUB
00023CAC	R-DSX$ENDSUB
00023E1D	R-DSX$CKLOOP
00023E5D	R-DSX$INLOOP
00023E84	R-CHR$MCHK
00023EEB	R-DSR$FAULT_CLEAR
00023EEF	R-EXE$ALLOCBUF
00023EF3	R-EXE$ALLOCCEB
00023EF7	R-EXE$ALLOCJIB
00023F00	R-EXE$ALLOCIRP
00023F04	R-EXE$ALLOCPCB
00023F0C	R-EXE$ALLOCPQB
00023F15	R-EXE$ALLOCTQE
00023F4F	R-EXE$ALONONPAGED
00023FB6	R-EXE$ALLOCATE
00023FDE	R-EXE$DEANONPAGED
00024051	R-EXE$DEALLOCATE
000240A7	R-MEMPOOL$INIT
0002417C	R-MAPMEM
0002432B	R-MAPMEM$IOSPACE
00024426	R-MAPFREE
000244C2	R-CRD$EXPREG
000244DF	R-EXE$EXPREG
000245E3	R-EXE$CNTREG

Value Symbols...

```
000246D0      R-DSX$MMON
000246DE      R-DSX$MMOFF
000246F8      R-DSR$MMENABLE
00024708      R-CMK$MMENABLE
00024719      R-EXE$SETPRT
0002472B      R-CMK$SETPRT
00024801      R-RPTEADR
00024843      R-MMG$IOLOCK
0002484D      R-MMG$UNLOCK
00024851      R-DSX$MAPDBGBLOCK
0002489F      R-DSX$FREEDBGSYM
000248C9      R-MAP_DEBUGGER
000248E2      R-UNMAP_DEBUGGER
0002497C      R-INI$RDONLY                    R-INI$WRITABLE
00024A7F      R-ODS$OPEN
00024B70      R-ODS$GET
00024C99      R-ODS$READ
00024D4C      R-DSP$CONFIG_ADP
00024E28      R-MAP$IOSPACE
00024E76      R-FETCH_LONG
00024EAA      R-IOC$PURGDATAP
00024EBD      R-UBADP_CPU
00024ECA      R-UBA$INITIAL
00024ECC      R-UBA$UNEXINT
00024ECD      R-IO$TESTCSR_L
00024EEC      R-IO$TESTCSR_W
00024F0B      R-IOGEN$CONN_VEC
00024F16      R-QIOCLEAN_CPU
00024F44      R-ONLY_SCB
00024F49      R-SCBVECTOR_000
00024F4D      R-SCBVECTOR_038
00024F51      R-SCBVECTOR_03C
00024F55      R-SCBVECTOR_050
00024F59      R-SCBVECTOR_054
00024F5D      R-SCBVECTOR_058
00024F61      R-SCBVECTOR_05C
00024F65      R-SCBVECTOR_060
00024F69      R-SCBVECTOR_064
00024F6D      R-SCBVECTOR_068
00024F71      R-SCBVECTOR_06C
00024F75      R-SCBVECTOR_070
00024F79      R-SCBVECTOR_074
00024F7D      R-SCBVECTOR_078
00024F81      R-SCBVECTOR_07C
00024F85      R-SCBVECTOR_080
00024F89      R-SCBVECTOR_0C4
00024F8D      R-SCBVECTOR_0C8
00024F91      R-SCBVECTOR_0CC
00024F95      R-SCBVECTOR_0D0
00024F99      R-SCBVECTOR_0D4
00024F9D      R-SCBVECTOR_0D8
00024FA1      R-SCBVECTOR_0DC
```

Value Symbols...

00024FA5 R-SCBVECTOR_0E0
00024FA9 R-SCBVECTOR_0E4
00024FAD R-SCBVECTOR_0E8
00024FB1 R-SCBVECTOR_0EC
00024FB5 R-SCBVECTOR_0F0
00024FB9 R-SCBVECTOR_0F4
00024FBD R-SCBVECTOR_0F8
00024FC1 R-SCBVECTOR_0FC
00024FCB R-ONLY_CLOCK_INIT
00024FD0 R-DSX$GETDATA
00025046 R-DSX$GETVIELD
000250A8 R-DSX$GETADDRESS
000250F2 R-DSX$GETLOGICAL
000251C4 R-DSX$GETSTRING
000254BA R-DSX$SUPERPARSE
000254C1 R-DSX$PARSE
0002568C R-SCAN$SYMBOL
0002568E R-DS$GT_VSYMBOL
0002569E R-SCAN$ALPHA
000256A0 R-DS$GT_VALPHA
000256B0 R-SCAN$ALPHANUM
000256B2 R-DS$GT_VALPHANUM
000256C2 R-SCAN$FILE
000256C4 R-DS$GT_VFILE
000256D4 R-SCAN$SPACE R-SCAN$SPACES
000256D6 R-DS$GT_VSPACE
000256E8 R-SCAN$SPANC
00025703 R-SCAN$COMPARE
00025725 R-DS$GT_VHEX
00025746 R-SCAN$DECIMAL
0002574B R-SCAN$OCTAL
00025750 R-SCAN$BINARY
00025755 R-SCAN$HEX
00025758 R-SCAN$NUMERIC
000257F4 R-SCAN$SEARCHLIST
00025843 R-SCAN$DEVICE
0002589F R-BREAKUP_FILE
00025938 R-DSX$LOADPCS
00025942 R-DSV$INITPCS
00025952 R-DSV$SETPAGE
00025979 R-DSV$SHOWPAGE
00025999 R-DSX$TYPE_OUT
00025C0D R-DSX$PRINT
00025C5B R-DSX$PRINTS
00025C68 R-DSX$PRINTX
00025C75 R-DSX$PRINTB
00025C8D R-DSX$PRINTF
00025CA2 R-DSX$PRINTI
00025DD4 R-DSX$CVTREG
00025F90 R-QA$NORMAL_START
0002605B R-QA$MULTIPLE_PASS
0002612A R-QA$LOOP_ON_TEST

Value Symbols...

```
0002622C      R-QA$LOOP_ON_SUBTEST
0002634C      R-QA$RUN_BACKWARDS
00026428      R-VRSETQA
000264D5      R-VRSHOWQA
00026560      R-DSX$BRANCH
000266DE      R-QA$ADD_FORCED_ERROR
000266E1      R-QA$ADD_QA_ERROR
0002687F      R-QA$FIND_USER_CALL_FRAME
000268FF      R-QA$MAIN
00026AD8      R-QA$PRINT_FORCED_ERRORS
00026B94      R-EXE$QIO
00026BDF      R-QIO$INITIALIZE
00026C33      R-QIO$CLEANUP
00026CFA      R-QIO$ADDUNIT
000271A0      R-EXE$MAXACMODE
000271AF      R-BUG$CHECK
000274E6      R-IOGEN$CNTRL_INI
00027574      R-QIO$DEALLOCATE
000275B9      R-EXE$PWRTIMCHK
000275CC      R-DSR$ALLOCATE_PSEUDO_RPB
00027645      R-DSR$DEALLOCATE_PSEUDO_RPB
00027658      R-DSX$CHECK_SUPPORT
000277F7      R-RMS$ERROR
00027889      R-RMS$LOAD_XAB
000278D8      R-RMS$ALLOC_CACHE
00027928      R-RMS$INIT
00027937      R-RMS$CLEANUP
000279BB      R-RMS$OPEN
00027C43      R-RMS$CONNECT
00027CC5      R-RMS$GET
00027D62      R-RMS$READ
00027DF1      R-RMS$DISCONNECT
00027E4B      R-RMS$CLOSE
00027F71      R-RT11$OPEN
00028067      R-RT11$GET
00028177      R-RT11$READ
000281E0      R-DSX$INITSCB
00028203      R-CMK$INITSCB
000282B7      R-DSX$SETVEC
00028372      R-DSX$CLRVEC
00028418      R-DSX$SETIPL
00028431      R-CMK$SETIPL
000284D0      R-SCRIPT$INIT
000284DA      R-SCRIPT$OPEN
00028621      R-SCRIPT$FLUSH
0002863C      R-SCRIPT$STOP
00028645      R-SCRIPT$CONT
0002867D      R-DS_SUPERLINE
00028684      R-DS_GETLINE
00028A4C      R-DSR$CHECKLOAD
00028A6B      R-DSV$SHOWMEMORY
00028E21      R-VRSETMEM
```

Value	Symbols...		
00028E7F	R-VRSTART		
00028ED0	R-VRRESTART		
00029228	R-DSX$ABORT		
0002924E	R-VRABORT		
00029301	R-VRSUPPORT		
0002935B	R-VRSHOWSECTIONS		
000293AD	R-DS$ENTRY		
000293B5	R-MT$CHECK_ACCESS		
000293B7	R-ERL$DEVICERR	R-ERL$DEVICTMO	R-ERL$RELEASEMB
	R-EXE$SNDEVMSG		
000293B8	R-EXE$BUFFRQUOTA	R-EXE$BUFQUOPRC	R-SCH$WAKE
000293BC	R-SCH$LOCKW		
000293C3	R-SCH$UNLOCK		
000293DD	R-LIB$CVT_DTB		
000293E4	R-LIB$CVT_OTB		
000293EB	R-LIB$CVT_HTB		
00029600	R-SCB_IMAGE		
00029700	R-BOO$AL_VECTOR		
00029714	R-EXE$GB_CPUTYPE		
00029715	R-EXE$GB_ADPTYPE		
00029716	R-BOO$QIO		
000297F1	R-BOO$MAP		
000298A2	R-BOO$DRIVER		
0002A000	R-UD_INIT		
0002A0C4	R-DS$GB_UDA50_INIT		
0002A250	R-UDA_RESPONSE		
0002A2E8	R-TS1T_DRIVER		
0002A600	R-TU_INIT		
0002AA72	R-BOO$SELECT		
0002AADC	R-ACP$ACCESS	R-ACP$ACCESSNET	
0002AAE7	R-ACP$DEACCESS		
0002AAF2	R-ACP$MODIFY		
0002AAFD	R-ACP$MOUNT		
0002AB08	R-ACP$READBLK		
0002AB25	R-ACP$WRITEBLK		
0002ABDC	R-EXE$ASSIGN		
0002AD08	R-IOC$GETBYTE		
0002AD19	R-IOC$PUTBYTE		
0002AD2A	R-IOC$INITBUFWIND		
0002AD31	R-IOC$MOVFRUSER		
0002AD35	R-IOC$MOVFRUSER2		
0002AD42	R-IOC$MOVFRUSER1		
0002AD49	R-IOC$MOVTOUSER		
0002AD4D	R-IOC$MOVTOUSER2		
0002AD5A	R-IOC$MOVTOUSER1		
0002AD61	R-IOC$FILSPI		
0002AD73	R-EXE$CANCEL		
0002AE78	R-COM$DELATTNAST		
0002AEAB	R-COM$FLUSHATTNS		
0002AEDB	R-COM$POST		
0002AEEB	R-COM$DRVDEALMEM		
0002AF10	R-COM$SETATTNAST		

Value	Symbols...
0002AFD5	R-EXE$ASCTIM
0002B076	R-EXE$BINTIM
0002B26C	R-EXE$NUMTIM
0002B386	R-EXE$DASSGN
0002B446	R-EXE$ALLOC
0002B4DF	R-EXE$DALLOC
0002B59A	R-DSX$MOUNT
0002B5B4	R-DSX$DISMOUNT
0002B5D4	R-DR$INT
0002B5EB	R-DR$INITIAL
0002B5F4	R-EXE$IOFORK
0002B5F9	R-EXE$FORK
0002B618	R-EXE$FORKDSPTH
0002B65C	R-EXE$GETTIM
0002B678	R-EXE$GTCHAN
0002B700	R-IOC$AL_SYSUCB
0002B708	R-IOC$GL_DEVLIST
0002B70C	R-IOC$GL_ADPLIST
0002B710	R-IOC$GL_DPTLIST
0002B718	R-IOC$CANCELIO
0002B72F	R-IOC$DIAGBUFILL
0002B760	R-IOC$RELSCHAN
0002B76A	R-IOC$RELCHAN
0002B7AD	R-IOC$REQSCHANH
0002B7B7	R-IOC$REQSCHANL
0002B7C1	R-IOC$REQPCHANH
0002B7CA	R-IOC$REQPCHANL
0002B7F5	R-IOC$REQCOM
0002B835	R-IOC$INITIATE
0002B868	R-IOC$RELDATAP
0002B8B8	R-IOC$REQDATAPNW
0002B8BC	R-IOC$REQDATAP
0002B8F5	R-IOC$RELMAPREG
0002B92F	R-IOC$REQMAPREG
0002B941	R-IOC$ALTUBAMAP
0002B95F	R-IOC$ALOUBAMAPN
0002B963	R-IOC$ALOUBAMAP
0002B9C5	R-IOC$RETURN
0002B9C6	R-IOC$WFIKPCH
0002B9E8	R-IOC$WFIRLCH
0002BA0C	R-IOC$ALLOSPT
0002BA21	R-IOC$CVT_DEVNAM
0002BA55	R-IOC$FFCHAN
0002BA7C	R-IOC$SEARCHDEV
0002BA81	R-IOC$SEARCHGEN
0002BB73	R-IOC$UNLOCK
0002BB77	R-IOC$VERIFYCHAN
0002BBAC	R-IOC$APPLYECC
0002BC20	R-IOC$CVTLOGPHY
0002BC47	R-IOC$MAPVBLK
0002BCDA	R-IOC$UPDATRANSP
0002BD16	R-IOC$SENSEDISK

Value	Symbols...		
0002BD24	R-IOC$LOADMBAMAP		
0002BD6E	R-IOC$LOADUBAMAPA		
0002BD85	R-IOC$LOADUBAMAP		
0002BE01	R-IOC$PTETOPFN		
0002BE24	R-MBA$INT		
0002BECF	R-MBA$INITIAL		
0002BED8	R-DSX$PROBE		
0002BF4B	R-ACC$HANDLER		
0002BF90	R-EXE$ONEPARM		
0002BF96	R-EXE$ZEROPARM		
0002BF9C	R-EXE$MODIFY		
0002BFA2	R-EXE$READ		
0002BFAB	R-EXE$WRITE		
0002BFCE	R-EXE$READLOCK		
0002BFD1	R-EXE$WRITELOCK		
0002BFD4	R-EXE$MODIFYLOCK		
0002BFD7	R-EXE$MODIFYLOCKR		
0002BFE1	R-EXE$READLOCKR		
0002BFE8	R-EXE$WRITELOCKR		
0002C05E	R-EXE$READCHK		
0002C064	R-EXE$WRITECHK		
0002C072	R-EXE$READCHKR		
0002C09C	R-EXE$WRITECHKR		
0002C0D1	R-EXE$SETCHAR		
0002C0EF	R-EXE$SETMODE		
0002C0FF	R-EXE$SENSEMODE		
0002C127	R-EXE$CARRIAGE		
0002C177	R-VMS$QIO		
0002C30A	R-EXE$ABORTIO		
0002C317	R-EXE$FINISHIOC		
0002C319	R-EXE$FINISHIO		
0002C331	R-EXE$QIODRVPKT		
0002C335	R-EXE$ALTQUEPKT		
0002C357	R-EXE$QIOACPPKT		
0002C369	R-EXE$QIORETURN		
0002C372	R-EXE$INSIOQ		
0002C38D	R-EXE$INSERTIRP		
0002C3A8	R-IOC$INITDRV		
0002C3AE	R-IOC$REINITDRV		
0002C448	R-EXE$UNWIND		
0002C600	R-AX_BUFF	R-PHD_BASE	
0002C678	R-PCB_BASE		
0002C800	R-DS$AX_JIB		
0002C86C	R-DS$AX_ARB		
0002C8E4	R-DS$AX_SOFTPCB		
0002CA00	R-SCB_BASE		
0002D200	R-SCB_UNKINT		
0002DA00	R-STACK_BASE		
0002F000	R-KSTKPTR		
0002F800	R-ISTKPTR		
0002FE00	R-ESTKPTR		
00030400	R-SSTKPTR		

Value		Symbols...	
00030A00	R-POPT_BASE		R-USTKPTR
00080000	DEV$M_MNT		
00200000	DEV$M_DMT		
00660000	IOC$K_DIAGPID		
01000000	DEV$M_FOR		
02000000	DEV$M_SWL		
03000000	DS$GK_SID		
40000000	DEV$M_RCK		IOC$K_IOSPACE
7FFEDEB0	SYS$CRELOG		
7FFEDEF0	SYS$DCLEXH		
7FFEE188	SYS$PUT		
7FFEE1C8	SYS$CREATE		
7FFEE218	SYS$SPACE		
7FFEE230	SYS$PARSE		
7FFEE248	SYS$SEARCH		
7FFEE250	SYS$SETDDIR		
7FFEE3F8	SYS$GETSYI		
80000000	DEV$M_WCK		
800000C8	SYS$CRETVA		
80000110	SYS$DELTVA		
80000208	SYS$SETEXV		

Key for special characters above:

*	- Undefined
U	- Universal
R	- Relocatable
X	- External

Image Synopsis

```
Virtual memory allocated:                   00010000 0002C7FF 0001C800 (116736. bytes, 228. pages)
Stack size:                                       20. pages
Image header virtual block limits:                 1.        1. (    1. block)
Image binary virtual block limits:                 2.      229. (  228. blocks)
Image name and identification:              ENSAA  7.0
Number of files:                                   2.
Number of modules:                                91.
Number of program sections:                       18.
Number of global symbols:                       1025.
Including undefined count of:                      2.
Number of cross references:                     2563.
Number of image sections:                          6.
User transfer address:                      000293AD
Image type:                                 EXECUTABLE.
Map format:                                 FULL WITH CROSS REFERENCE in file DRB1:[MUSE.WORK.MODULES]ENSAA.MAP;574
Estimated map length:                       321. blocks
```

Link Run Statistics

Performance Indicators	Page Faults	CPU Time	Elapsed Time
Command processing:	88	00:00:00.49	00:00:00.99
Pass 1:	372	00:00:10.81	00:00:29.96
Allocation/Relocation:	77	00:00:00.64	00:00:01.07
Pass 2:	233	00:00:11.18	00:00:26.01
Map data after object module synopsis:	16	00:00:07.08	00:00:07.60
Symbol table output:	0	00:00:00.02	00:00:00.16
Total run values:	786	00:00:30.22	00:01:05.79

```
Using a working set limited to 1850 pages and 702 pages of data storage (excluding image)

Total number object records read (both passes):   10796
    of which 5398 were in libraries and 3294 were DEBUG data records containing 1234231 bytes

Number of modules extracted explicitly             = 86
    with 5 extracted to resolve undefined symbols

2 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

LINK/MAP=WORK$:ENSAA;574/CROSS/FULL/NODEBUG/NOTRACEBACK/NOSYSSHR/EXE=WORK$:ENSAA.EXE;574 DRB1:[DS.WORK]ENSAA.OPT;/OPTIONS
```

IDENTIFICATION

PRODUCT CODE: ZZ-ENSAA-7.0

PRODUCT TITLE: VAX DIAGNOSTIC SUPERVISOR

PRODUCT DATE: 18-JUN-1984

DEPARTMENT: BASE SYSTEM DIAGNOSTIC ENGINEERING

Table of contents

(1) 87 ACCESS AND CREATE ACP FUNCTION PROCESSING
(1) 136 DEACCESS ACP FUNCTION PROCESSING
(1) 190 DELETE AND MODIFY ACP FUNCTION PROCESSING
(1) 222 MOUNT ACP FUNCTION PROCESSING
(1) 260 READ AND WRITE BLOCK ACP FUNCTION PROCESSING

```
                                .TITLE  ACPFDT  *** ACPFDT ACP function dispatch
                                .IDENT  /06-03/
                                .PSECT  SEP, SHR, EXE, WRT, LONG

                                ;*****************************************************************************
                                ;*                                                                           *
                                ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                                ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                                ;*                                                                           *
                                ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                                ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
                                ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
                                ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                                ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                                ;*  TRANSFERRED.                                                             *
                                ;*                                                                           *
                                ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                                ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                                ;*  CORPORATION.                                                             *
                                ;*                                                                           *
                                ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                                ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                                ;*                                                                           *
                                ;*****************************************************************************

                                ; D. N. CUTLER 22-NOV-76

                                ; MODIFIED BY:

                                ;       V0006   ACG0100         Andrew C. Goldstein,    3-Jan-1980  17:19
                                ;               Permit partial dismount of volume sets
                                ;
                                ;       V0005   RIH0036         Richard I. Hustvedt     02-Nov-1979
                                ;               Set IRP$V_FILACP for file acp requests and bump DIO count
                                ;
                                ;       V0004   RIH0033         Richard I. Hustvedt     16-Oct-1979     15:55
                                ;               Change PCB$W_BYTCNT to JIB$L_BYTCNT; PCB$W_BYTLM to JIB$L_BYTLM;
                                ;               PCB$W_FILCNT to JIB$W_FILCNT.
                                ;
                                ;       V0003   ACG0047         Andrew C. Goldstein,    8-Aug-1979  17:23
                                ;               Interface change to protection check routines
                                ;
                                ;       V0002   ACG23542        Andrew C. Goldstein,  4-May-1979  14:43
                                ;       Check LBN of mapped virtual I/O against UCB$L_MAXBLOCK
                                ;
02                              ;
02                              ;               Dave Butenhof   13-may-1980, Version 5.4
02                              ;       02      Modify VMS V2.0 source for Supervisor environment.
02                              ;
03                              ;
03                              ;               Dave Butenhof, 08-oct-1980, version 6.1
03                              ;       03      Changed some word offsets to longword.
                                ;**
                                ;
                                ; ACP FUNCTION DECISION TABLE ACTION ROUTINES
                                ;
```

```
                                0000     50 ; MACRO LIBRARY CALLS
                                0000     51 ;
                                0000     52
                                0000     53         $CCBDEF                          ;DEFINE CCB OFFSETS
                                0000     54         $DEVDEF                          ;DEFINE DEVICE CHARACTERISTICS BITS
                                0000     55         $DYNDEF                          ;DEFINE STRUCTURE TYPE CODES
                                0000     56         $IPLDEF                          ;DEFINE INTERRUPT PRIORITY LEVELS
                                0000     57         $IODEF                           ;DEFINE I/O FUNCTION CODES
                                0000     58         $IRPDEF                          ;DEFINE IRP OFFSETS
                                0000     59         $JIBDEF                          ;DEFINE JIB OFFSETS
                                0000     60         $PCBDEF                          ;DEFINE PCB OFFSETS
                                0000     61         $PRDEF                           ;DEFINE PROCESSOR REGISTERS
                                0000     62         $PRVDEF                          ;DEFINE PRIVILEGE BITS
                                0000     63         $UCBDEF                          ;DEFINE UCB OFFSETS
                                C000     64         $VADEF                           ;DEFINE VIRTUAL ADDRESS FIELDS
                                0000     65         $VCBDEF                          ;DEFINE VCB OFFSETS
                                0000     66         $WCBDEF                          ;DEFINE WCB OFFSETS
                                0000     67
                                0000     68 ;
                                0000     69 ; LOCAL SYMBOLS
                                0000     70 ;
                                0000     71 ; ARGUMENT LIST OFFSET DEFINITIONS
                                0000     72 ;
                                0000     73
                    00000000    0000     74 P1=0                                     ;FIRST FUNCTION DEPENDENT PARAMETER
                    00000004    0000     75 P2=4                                     ;SECOND FUNCTION DEPENDENT PARAMETER
                    00000008    0000     76 P3=8                                     ;THIRD FUNCTION DEPENDENT PARAMETER
                    0000000C    0000     77 P4=12                                    ;FOURTH FUNCTION DEPENDENT PARAMETER
                    00000010    0000     78 P5=16                                    ;FIFTH FUNCTION DEPENDENT PARAMETER
                    00000014    0000     79 P6=20                                    ;SIXTH FUNCTION DEPENDENT PARAMETER
                                0000     80
                                0000     81 ;
                                0000     82 ; MAXIMUM NUMBER OF ACP DESCRIPTORS
                                0000     83 ;
                                0000     84
                    00000016    0000     85 MXDESCR=5+17                             ;MAXIMUM NUMBER OF ACP DESCRIPTORS
```

```
                                0000     87          .SBTTL  ACCESS AND CREATE ACP FUNCTION PROCESSING
                                0000     88 ;+
                                0000     89 ; ACP$ACCESS - ACCESS AND CREATE ACP FUNCTION PROCESSING
                                0000     90 ; ACP$ACCESSNET - ACCESS (CONNECT) TO NETWORK FUNCTION PROCESSING
                                0000     91 ;
                                0000     92 ; ***TBS***
                                0000     93 ;
                                0000     94 ; INPUTS:
                                0000     95 ;
                                0000     96 ;       R0 = SCRATCH.
                                0000     97 ;       R1 = SCRATCH.
                                0000     98 ;       R2 = SCRATCH.
                                0000     99 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                0000    100 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                C000    101 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                0000    102 ;       R6 = ADDRESS OF CCB.
                                0000    103 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                0000    104 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                0000    105 ;       R9 = SCRATCH.
                                0000    106 ;       R10 = SCRATCH.
                                0000    107 ;       R11 = SCRATCH.
                                0000    108 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                0000    109 ;
                                0000    110 ; OUTPUTS:
                                0000    111 ;
                                0000    112 ; ***TBS***
                                0000    113 ;-
                                0000    114
                                0000    115          .ENABL  LSB
-2                              0000    118 ACP$ACCESS::                            ;ACCESS AND CREATE ACP FUNCTION PROCESSING
-1                              0000    120 ACP$ACCESSNET::                         ;ACCESS (CONNECT) TO NETWORK FUNCTION PROCES
02              50    0000'8F   3C  0000     .1          MOVZWL  #SS$_FILNOTACC,R0   ; ROUTINE NOT IMPLEMENTED IN S/A
03               00000000'EF    16  0005     .2          JSB     L^EXE$ABORTIO       ; ABORT THIS QIO
```

```
                                000B   136          .SBTTL  DEACCESS ACP FUNCTION PROCESSING
                                000B   137 ;+
                                000B   138 ; ACP$DEACCESS - DEACCESS ACP FUNCTION PROCESSING
                                000B   139 ;
                                000B   140 ; ***TBS***
                                000B   141 ;
                                000B   142 ; INPUTS:
                                000B   143 ;
                                000B   144 ;       R0 = SCRATCH.
                                000B   145 ;       R1 = SCRATCH.
                                000B   146 ;       R2 = SCRATCH.
                                000B   147 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                000B   148 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                000B   149 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                C00B   150 ;       R6 = ADDRESS OF CCB.
                                000B   151 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                000B   152 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                000B   153 ;       R9 = SCRATCH.
                                000B   154 ;       R10 = SCRATCH.
                                000B   155 ;       R11 = SCRATCH.
                                000B   156 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                000B   157 ;
                                000B   158 ; OUTPUTS:
                                000B   159 ;
                                000B   160 ; ***TBS***
                                000B   161 ;-
                                000B   162
                                000B   163 ACP$DEACCESS::                               ;DEACCESS ACP FUNCTION PROCESSING
02                       50    0000'8F    3C  000B     .1          MOVZWL  #SS$_FILNOTACC,R0       ; FUNCTION NOR SUPPORTED BY SUPERVISOR
03                        00000000'EF    16  0010     .2          JSB     L^EXE$ABORTIO           ; ABORT THIS QIO
```

```
                     0016   190          .SBTTL  DELETE AND MODIFY ACP FUNCTION PROCESSING
                     0016   191 ;+
                     0016   192 ; ACP$MODIFY - DELETE AND MODIFY ACP FUNCTION PROCESSING
                     0016   193 ;
                     0016   194 ; ***TBS***
                     0016   195 ;
                     0016   196 ; INPUTS:
                     0016   197 ;
                     0016   198 ;       R0 = SCRATCH.
                     0016   199 ;       R1 = SCRATCH.
                     0016   200 ;       R2 = SCRATCH.
                     0016   201 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     0016   202 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0016   203 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     C016   204 ;       R6 = ADDRESS OF CCB.
                     0016   205 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     0016   206 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     0016   207 ;       R9 = SCRATCH.
                     0016   208 ;       R10 = SCRATCH.
                     0016   209 ;       R11 = SCRATCH.
                     0016   210 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     0016   211 ;
                     0016   212 ; OUTPUTS:
                     0016   213 ;
                     0016   214 ; ***TBS***
                     0016   215 ;-
                     0016   216
                     0016   217 ACP$MODIFY::                                ;DELETE AND MODIFY ACP FUNCTION PROCESSING
02                   0016    .1          MOVZWL  #SS$_FILNOTACC,R0         ; ROUTINE NOT SUPPORTED BY SUPERVISOR
03                   0016    .2          JSB     L^EXE$ABORTIO             ; ABORT THIS QIO
```

```
                        0021   222          .SBTTL  MOUNT ACP FUNCTION PROCESSING
                        0021   223 ;+
                        0021   224 ; ACP$MOUNT - MOUNT ACP FUNCTION PROCESSING
                        0021   225 ;
                        0021   226 ; ***TBS***
                        0021   227 ;
                        0021   228 ; INPUTS:
                        0021   229 ;
                        0021   230 ;       R0 = SCRATCH.
                        0021   231 ;       R1 = SCRATCH.
                        0021   232 ;       R2 = SCRATCH.
                        0021   233 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        0021   234 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        0021   235 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                        0021   236 ;       R6 = ADDRESS OF CCB.
                        0021   237 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                        0021   238 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                        0021   239 ;       R9 = SCRATCH.
                        0021   240 ;       R10 = SCRATCH.
                        0021   241 ;       R11 = SCRATCH.
                        0021   242 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                        0021   243 ;
                        0021   244 ; OUTPUTS:
                        0021   245 ;
                        0021   246 ; ***TBS***
                        0021   247 ;-
                        0021   248
                        0021   249 ACP$MOUNT::                                 ;MOUNT ACP FUNCTION PROCESSING
02           50  0000'8F   3C   0021    .1          MOVZWL  #SS$_FILNOTACC,R0  ; FUNCTION NOT SUPPORTED BY SUPERVISOR
03            00000000'EF  16   0026    .2          JSB     L^EXE$ABORTIO      ; ABORT THIS QIO
-8                         002C   258               .DSABL  LSB
```

```
                                002C   260          .SBTTL  READ AND WRITE BLOCK ACP FUNCTION PROCESSING
                                002C   261 ;+
                                002C   262 ; ACP$READBLK - READ BLOCK ACP FUNCTION PROCESSING
                                002C   263 ; ACP$WRITEBLK - WRITE BLOCK ACP FUNCTION PROCESSING
                                002C   264 ;
                                002C   265 ; ***TBS***
                                002C   266 ;
                                002C   267 ; INPUTS:
                                002C   268 ;
                                002C   269 ;       R0 = SCRATCH.
                                002C   270 ;       R1 = SCRATCH.
                                002C   271 ;       R2 = SCRATCH.
                                002C   272 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                002C   273 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                C02C   274 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                002C   275 ;       R6 = ADDRESS OF CCB.
                                002C   276 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                002C   277 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                002C   278 ;       R9 = SCRATCH.
                                002C   279 ;       R10 = SCRATCH.
                                002C   280 ;       R11 = SCRATCH.
                                002C   281 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                002C   282 ;
                                002C   283 ; OUTPUTS:
                                002C   284 ;
                                002C   285 ; ***TBS***
                                002C   286 ;-
                                002C   287
                                002C   288          .ENABL  LSB
                                002C   289 ACP$READBLK::                                ;READ BLOCK ACP FUNCTION PROCESSING
                 5A   0000'CF   9E   002C   290          MOVAB   W^EXE$READLOCK,R10      ;SET ADDRESS OF READ CHECK AND LOCK ROUTINE
-1            22 34 A5    1E   E1   0031   292          BBC     S^#DEV$V_RCK,UCB$L_DEVCHAR(R5),20$ ;IF CLR, NO DEVICE READ CHECK
                 4000 8F    A8   0036   293 10$:     BISW    #IO$M_DATACHECK,IRP$W_FUNC(R3) ;SET DATA CHECK ENABLE
                    20 A3         003A
                       1A   11   003C   294          BRB     20$                     ;
                                003E   295 ;
                                003E   296 ; ZERO LENGTH TRANSFER
                                003E   297 ;
                                003E   298
                  50   0000'8F   3C   003E   299 15$:     MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION
03                 00000000'EF   16   0043     .1          JSB     L^EXE$FINISHIOC         ;FINISH I/O OPERATION
-1                                     0049   301
                                0049   302 ACP$WRITEBLK::                               ;WRITE BLOCK ACP FUNCTION PROCESSING
                46 34 A5    19   E0   0049   303          BBS     S^#DEV$V_SWL,UCB$L_DEVCHAR(R5),60$ ;IF SET, SOFTWARE WRITE LOCKED
                 5A   0000'CF   9E   004E   304          MOVAB   W^EXE$WRITELOCK,R10     ;SET ADDRESS OF WRITE CHECK AND LOCK ROUTINE
-1              DE 34 A5    1F   E0   0053   306          BBS     S^#DEV$V_WCK,UCB$L_DEVCHAR(R5),10$ ;IF SET, DEVICE LEVEL WRITE CHECK
                    50    6C   D0   0058   307 20$:     MOVL    P1(AP),R0               ;GET STARTING ADDRESS OF TRANSFER
                   51   04 AC   3C   005B   308          MOVZWL  P2(AP),R1               ;GET NUMBER OF BYTES TO TRANSFER
                       DD   13   005F   309          BEQL    15$                     ;IF EQL ZERO LENGTH TRANSFER
                05 34 A5    05   E1   0061   310          BBC     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),30$ ;IF CLR, NOT SEQUENTIAL DEVICE
                    51    0E   D1   0066   311          CMPL    #14,R1                  ;RECORD TOO SHORT?
                          1B   14   0069   312          BGTR    45$                     ;IF GTR YES
                          6A   16   0068   313 30$:     JSB     (R10)                   ;CHECK BUFFER AND LOCK IN MEMORY
                   50   08 AC   D0   006D   314          MOVL    P3(AP),R0               ;GET STARTING MEDIA ADDRESS
                   51   32 A3   3C   0071   315          MOVZWL  IRP$W_BCNT(R3),R1       ;RETRIEVE TRANSFER BYTE COUNT
                3E A3    51    B0   0075   316          MOVW    R1,IRP$W_OBCNT(R3)      ;SET ORIGNINAL BYTE COUNT
                   3C A3         B4   0079   317          CLRW    IRP$W_ABCNT(R3)         ;CLEAR ACCUMULATED BYTE COUNT
```

```
                    06   00    ED   007C   318          CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;LOGICAL OR PHYSICAL I/O FUNCTION?
                 2F   20 A3                007F   319                  IRP$W_FUNC(R3),#IO$_LOGICAL ;
                         1B    15   0082   320          BLEQ    120$                        ;IF LEQ YES
02                       07    11   0084    .1          BRB     50$                         ; Supervisor does not support Virtual
-13                                 0086   334
                                    0086   335 ;
                                    0086   336 ; RECORD LENGTH TOO SHORT FOR MAGTAPE TRANSFER
                                    0086   337 ;
                                    0086   338
                    50   0000'8F    3C   0086   339 45$:     MOVZWL  #SS$_BADPARAM,R0            ;SET BAD PARAMETER STATUS
                         0C    11   008B   340          BRB     70$                         ;
                                    008D   341
                                    008D   342 ;
                                    008D   343 ; FILE NOT ACCESSED ON CHANNEL
                                    C08D   344 ;
                                    008D   345
                    50   0000'8F    3C   008D   346 50$:     MOVZWL  #SS$_FILNOTACC,R0           ;SET FILE NOT ACCESSED
                         05    11   0092   347          BRB     70$                         ;
                                    0094   348
                                    0094   349 ;
                                    0094   350 ; PRIVILEGE VIOLATION
                                    0094   351 ;
                                    0094   352
                    50   0000'8F    3C   0094   353 60$:     MOVZWL  #SS$_NOPRIV,R0              ;SET NO PRIVILEGE
03                00000000'EF    16   0099    .1 70$:     JSB     L^EXE$ABORTIO               ;
-1                                  009F   355
-22                   34 A3   50    D0   009F   378 120$:   MOVL    R0,IRP$L_MEDIA(R3)          ;SAVE MEDIA ADDRESS
                         59   50    7D   00A3   379          MOVQ    R0,R9                       ;SAVE TRANSFER PARAMETERS
                    50   0084 C4    D0   00A6   380          MOVL    PCB$L_ARB(R4),R0            ;GET ACCESS RIGHTS BLOCK
                      51   1A A5    3C   00AB   381          MOVZWL  UCB$W_VPROT(R5),R1          ;GET VOLUME PROTECTION MASK
                      52   1C A5    D0   00AF   382          MOVL    UCB$L_OWNUIC(R5),R2         ;GET VOLUME OWNER UIC
-2                       06   00    ED   00B3   385          CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;PHYSICAL I/O FUNCTION?
                      1F   20 A3          00B6   386                  IRP$W_FUNC(R3),#IO$_PHYSICAL ;
                              28    15   00B9   387          BLEQ    150$                        ;IF LEQ YES
                              15    A2   00BB   388          SUBW    #IO$_WRITELBLK-IO$_WRITEPBLK,- ;CONVERT TO PHYSICAL I/O FUNCTION
                           20 A3          00BD   389                  IRP$W_FUNC(R3)              ;
                   19 34 A5    05    E0   00BF   390          BBS     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),140$ ;IF SET, SEQUENTIAL DEVICE
                              5A    D7   00C4   391 130$:   DECL    R10                         ;ROUND BYTE COUNT DOWN
                      5A   F7 8F    78   00C6   392          ASHL    #-VA$S_BYTE,R10,R10         ;CONVERT TO BLOCK COUNT
                              5A         00CA
                         5A   59    C0   00CB   393          ADDL    R9,R10                      ;CALCULATE ENDING BLOCK NUMBER
                              19    1F   00CE   394          BCS     170$                        ;IF CS ILLEGAL BLOCK NUMBER
                    0084 C5   5A    D1   00D0   395          CMPL    R10,UCB$L_MAXBLOCK(R5)      ;LEGAL BLOCK NUMBER?
                              12    1E   00D5   396          BGEQU   170$                        ;IF GEQU NO
                         50   59    D0   00D7   397          MOVL    R9,R0                       ;RETRIEVE STARTING MEDIA ADDRESS
                            FF23'   30   00DA   398          BSBW    IOC$CVTLOGPHY               ;CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                         1800 8F    AA   00DD   399 140$:   BICW    #IO$M_INHERLOG!IO$M_INHSEEK,- ;CLEAR INHIBIT ERROR LOGGING
                           20 A3          00E1   400                  IRP$W_FUNC(R3)              ;AND EXPLICIT SEEK
03                 00000000'EF    16   00E3    .1 150$:   JSB     L^EXE$QIODRVPKT             ;QUEUE I/O PACKET TO DRIVER
-5                 50   0000'8F    3C   00E9   406 170$:   MOVZWL  #SS$_ILLBLKNUM,R0           ;SET ILLEGAL BLOCK NUMBER STATUS
                              05    11   00EE   407          BRB     190$
                    50   0000'8F    3C   00F0   408 180$:   MOVZWL  #SS$_ENDOFFILE,R0           ;SET END OF FILE STATUS
03                 00000000'EF    16   00F5    .1 190$:   JSB     L^EXE$FINISHIOC             ;FINISH I/O OPERATION
-1                                  00FB   410          .DSABL  LSB
```

```
-230                            00FB   642          .END
```

```
ACP$ACCESS          00000000 RG D  01      IRP$Q_NT_PRVMSK  0000003C  D      PCB$W_GRP          0000008A  D
ACP$ACCESSNET       00000000 RG D  01      IRP$S_FCODE    = 00000006  D      PCB$W_MEM          00000088  D
ACP$DEACCESS        0000000B RG D  01      IRP$V_FCODE    = 00000000  D      PCB$W_MTXCNT       0000000E  D
ACP$MODIFY          00000016 RG D  01      IRP$W_ABCNT      0000003C  D      PCB$W_PPGCNT       00000036  D
ACP$MOUNT           00000021 RG D  01      IRP$W_BCNT       00000032  D      PCB$W_PRCCNT       00000042  D
ACP$READBLK         0000002C RG D  01      IRP$W_BOFF       00000030  D      PCB$W_SIZE         00000008  D
ACP$WRITEBLK        00000049 RG D  01      IRP$W_CHAN       00000028  D      PCB$W_STATE        0000002C  D
CCB$B_AMOD          00000009    D          IRP$W_FUNC       00000020  D      PCB$W_TMBU         00000032  D
CCB$B_STS           00000008    D          IRP$W_OBCNT      0000003E  D      SIZ...           = 00000001  D
CCB$C_LENGTH        00000010    D          IRP$W_SIZE       00000008  D      SS$_BADPARAM       ********  X  01
CCB$K_LENGTH        00000010    D          IRP$W_STS        0000002A  D      SS$_ENDOFFILE      ********  X  01
CCB$L_DIRP          0000000C    D          IRP$W_TT_PRMPT   0000003C  D      SS$_FILNOTACC      ********  X  01
CCB$L_UCB           00000000    D          MXDESCR        = 00000016  D      SS$_ILLBLKNUM      ********  X  01
CCB$L_WIND          00000004    D          P1             = 00000000  D      SS$_NOPRIV         ********  X  01
CCB$W_IOC           0000000A    D          P2             = 00000004  D      SS$_NORMAL         ********  X  01
DEV$V_RCK         = 0000001E    D          P3             = 00000008  D      UCB$B_AMOD         00000053  D
DEV$V_SQD         = 00000005    D          P4             = 0000000C  D      UCB$B_CEX          00000077  D
DEV$V_SWL         = 00000019    D          P5             = 00000010  D      UCB$B_CM1          0000004A  D
DEV$V_WCK         = 0000001F    D          P6             = 00000014  D      UCB$B_CM2          0000004B  D
EXE$ABORTIO           ********  X  01      PCB$B_ASTACT     0000000C  D      UCB$B_DEVCLASS     00000038  D
EXE$FINISHIOC         ********  X  01      PCB$B_ASTEN      0000000D  D      UCB$B_DEVTYPE      00000039  D
EXE$QIODRVPKT         ********  X  01      PCB$B_PRI        0000000B  D      UCB$B_DIPL         00000052  D
EXE$READLOCK          ********  X  01      PCB$B_PRIB       0000002F  D      UCB$B_DX_SCTCNT    000000A6  D
EXE$WRITELOCK         ********  X  01      PCB$B_TYPE       0000000A  D      UCB$B_ERTCNT       00000070  D
IO$M_DATACHECK  = 00004000    D          PCB$B_WEFC       0000002E  D      UCB$B_ERTMAX       00000071  D
IO$M_INHERLOG   = 00000800    D          PCB$C_LENGTH     0000008C  D      UCB$B_FEX          00000076  D
IO$M_INHSEEK    = 00001000    D          PCB$K_LENGTH     0000008C  D      UCB$B_FIPL         0000000B  D
IO$_LOGICAL     = 0000002F    D          PCB$L_ARB        00000084  D      UCB$B_LOCSRV       0000003C  D
IO$_PHYSICAL    = 0000001F    D          PCB$L_ASTQBL     00000014  D      UCB$B_OFFNDX       00000094  D
IO$_WRITELBLK   = 00000020    D          PCB$L_ASTQFL     00000010  D      UCB$B_OFFRTC       00000095  D
IO$_WRITEPBLK   = 0000000B    D          PCB$L_EFC2P      00000058  D      UCB$B_REMSRV       0000003D  D
IOC$CVTLOGPHY         ********  X  01      PCB$L_EFC3P      0000005C  D      UCB$B_SECTORS      0000003C  D
IRP$B_CARCON        00000038    D          PCB$L_EFCS       00000050  D      UCB$B_SLAVE        00000074  D
IRP$B_EFN           00000022    D          PCB$L_EFCU       00000054  D      UCB$B_SPR          00000075  D
IRP$B_PRI           00000023    D          PCB$L_EFWM       0000004C  D      UCB$B_STATE        00000052  D
IRP$B_RMOD          0000000B    D          PCB$L_JIB        00000078  D      UCB$B_TRACKS       0000003D  D
IRP$B_TYPE          0000000A    D          PCB$L_OWNER      0000001C  D      UCB$B_TT_CRFILL    0000009D  D
IRP$C_LENGTH        0000005C    D          PCB$L_PHD        00000064  D      UCB$B_TT_DECRF     000000A1  D
IRP$K_LENGTH        0000005C    D          PCB$L_PHYPCB     00000018  D      UCB$B_TT_DELFF     000000A2  D
IRP$L_ARB           00000050    D          PCB$L_PID        00000060  D      UCB$B_TT_DESPEE    000000A0  D
IRP$L_AST           00000010    D          PCB$L_PQB        0000004C  D      UCB$B_TT_DETYPE    000000A4  D
IRP$L_ASTPRM        00000014    D          PCB$L_SQBL       00000004  D      UCB$B_TT_LFFILL    0000009E  D
IRP$L_DIAGBUF       00000044    D          PCB$L_SQFL       00000000  D      UCB$B_TT_SPEED     0000009C  D
IRP$L_EXTEND        0000004C    D          PCB$L_STS        00000024  D      UCB$B_TYPE         0000000A  D
IRP$L_IOQBL         00000004    D          PCB$L_UIC        00000088  D      UCB$B_VERTSZ       0000003F  D
IRP$L_IOQFL         00000000    D          PCB$L_WSSWP      00000020  D      UCB$C_LENGTH       00000074  D
IRP$L_IOSB          00000024    D          PCB$L_WTIME      00000028  D      UCB$C_MB_LENGTH    00000090  D
IRP$L_IOST1         00000034    D          PCB$Q_PRIV       0000007C  D      UCB$C_TT_LENGTH    000000BC  D
IRP$L_IOST2         00000038    D          PCB$T_LNAME      00000068  D      UCB$K_LENGTH       00000074  D
IRP$L_MEDIA         00000034    D          PCB$T_TERMINAL   00000044  D      UCB$K_MB_LENGTH    00000090  D
IRP$L_PID           0000000C    D          PCB$W_APTCNT     00000030  D      UCB$K_TT_LENGTH    000000BC  D
IRP$L_SEGVBN        00000040    D          PCB$W_ASTCNT     00000038  D      UCB$L_AMB          00000054  D
IRP$L_SEQNUM        00000048    D          PCB$W_BIOCNT     0000003A  D      UCB$L_ASTQBL       00000010  D
IRP$L_SVAPTE        0000002C    D          PCB$W_BIOLM      0000003C  D      UCB$L_ASTQFL       0000000C  D
IRP$L_TT_TERM       00000038    D          PCB$W_DIOCNT     0000003E  D      UCB$L_CPID         0000005C  D
IRP$L_UCB           0000001C    D          PCB$W_DIOLM      00000040  D      UCB$L_CRB          00000020  D
IRP$L_WIND          00000018    D          PCB$W_GPGCNT     00000034  D      UCB$L_DDB          00000024  D
```

```
UCB$L_DEVCHAR   00000034  D     UCB$W_EC2       00000092  D     VCB$L_USRLBLAST  00000044  D
UCB$L_DEVDEPEND 0000003C  D     UCB$W_ERRCNT    00000072  D     VCB$L_VPBL       00000040  D
UCB$L_DPC       00000080  D     UCB$W_FUNC      0000007E  D     VCB$L_VPFL       0000003C  D
UCB$L_DUETIM    0000005C  D     UCB$W_MB_SEED   00000000  D     VCB$L_WCB        00000038  D
UCB$L_DX_BFPNT  0000009C  D     UCB$W_MSGCNT    00000016  D     VCB$T_QNAME      0000000C  D
UCB$L_DX_BUF    00000098  D     UCB$W_MSGMAX    00000014  D     VCB$T_VOLNAME    00000014  D
UCB$L_DX_RXDB   000000A0  D     UCB$W_NT_CHAN   0000007C  D     VCB$W_CLUSTER    0000003C  D
UCB$L_EMB       00000078  D     UCB$W_OFFSET    0000008A  D     VCB$W_CUR_NUM    00000024  D
UCB$L_FIRST     00000014  D     UCB$W_REFC      00000050  D     VCB$W_CUR_SEQ    00000026  D
UCB$L_FPC       0000000C  D     UCB$W_SIZE      00000008  D     VCB$W_EXTEND     0000003E  D
UCB$L_FQBL      00000004  D     UCB$W_SRCADDR   0000001A  D     VCB$W_FILEPROT   0000004A  D
UCB$L_FQFL      00000000  D     UCB$W_STS       00000058  D     VCB$W_MCOUNT     0000004C  D
UCB$L_FR3       00000010  D     UCB$W_TT_DESIZE 000000A5  D     VCB$W_MODE       0000002C  D
UCB$L_FR4       00000014  D     UCB$W_UNIT      00000048  D     VCB$W_PENDERR    00000062  D
UCB$L_IOQBL     00000044  D     UCB$W_VPROT     0000001A  D     VCB$W_QUOSIZE    00000060  D
UCB$L_IOQFL     00000040  D     VA$S_BYTE     = 00000009  D     VCB$W_RECORDSZ   00000050  D
UCB$L_IRP       0000004C  D     VCB$B_BLOCKFACT 00000052  D     VCB$W_RVN        0000000E  D
UCB$L_LINK      0000002C  D     VCB$B_CUR_RVN   0000002F  D     VCB$W_SIZE       00000008  D
UCB$L_LOGADR    00000064  D     VCB$B_EOFDELTA  0000004E  D     VCB$W_START_NUM  00000028  D
UCB$L_MAXBLOCK  00000084  D     VCB$B_IBMAPSIZE 00000038  D     VCB$W_START_SEQ  0000002A  D
UCB$L_MB_MBX    0000007C  D     VCB$B_IBMAPVBN  0000003A  D     VCB$W_TRANS      0000000C  D
UCB$L_MB_PORT   0C00008C  D     VCB$B_LRU_LIM   00000049  D     WCB$B_ACCESS     0000000B  D
UCB$L_MB_RAST   00000078  D     VCB$B_QNAMECNT  0000000B  D     WCB$B_TYPE       0000000A  D
UCB$L_MB_SHB    00000080  D     VCB$B_RESFILES  0000004F  D     WCB$C_LENGTH     00000024  D
UCB$L_MB_WAST   00000074  D     VCB$B_SBMAPSIZE 00000039  D     WCB$C_MAP        00000024  D
UCB$L_MB_WIOQBL 00000088  D     VCB$B_SBMAPVBN  0J000038  D     WCB$K_LENGTH     00000024  D
UCB$L_MB_WIOQFL 00000084  D     VCB$B_STATUS    0000000B  D     WCB$K_MAP        00000024  D
UCB$L_MEDIA     0000008C  D     VCB$B_STATUS2   00000053  D     WCB$L_FCB        00000018  D
UCB$L_NT_DATSSB 00000074  D     VCB$B_TM        0000002E  D     WCB$L_LBN        00000002  D
UCB$L_NT_INTSSB 00000078  D     VCB$B_TYPE      0000000A  D     WCB$L_ORGUCB     00000010  D
UCB$L_OPCNT     00000060  D     VCB$B_WINDOW    00000048  D     WCB$L_P1_LBN     00000026  D
UCB$L_OWNUIC    0000001C  D     VCB$C_COMLEN    00000024  D     WCB$L_P2_LBN     0000002C  D
UCB$L_PID       00000028  D     VCB$C_LENGTH    00000064  D     WCB$L_PID        0000000C  D
UCB$L_RQBL      00000004  D     VCB$C_MRKLEN    0000000B  D     WCB$L_PREVLBN    FFFFFFFC  D
UCB$L_RQFL      00000000  D     VCB$K_COMLEN    00000024  D     WCB$L_RVT        0000001C  D
UCB$L_SVAPTE    00000068  D     VCB$K_LENGTH    00000064  D     WCB$L_STVBN      00000020  D
UCB$L_SVPN      00000064  D     VCB$K_MRKLEN    0000000B  D     WCB$L_WLBL       00000004  D
UCB$L_TT_DECHAR 000000A8  D     VCB$L_AQB       00000010  D     WCB$L_WLFL       00000000  D
UCB$L_TT_RDUE   0000008C  D     VCB$L_BLOCKBL   00000004  D     WCB$W_ACON       00000014  D
UCB$L_TT_RTIMOU 000000B8  D     VCB$L_BLOCKFL   00000000  D     WCB$W_COUNT      00000000  D
UCB$L_VCB       00000030  D     VCB$L_CACHE     00000058  D     WCB$W_NMAP       00000016  D
UCB$T_PARTNER   0000000C  D     VCB$L_CUR_FID   00000024  D     WCB$W_P1_COUNT   00000024  D
UCB$W_BCNT      0000006E  D     VCB$L_FCBBL     00000004  D     WCB$W_P2_COUNT   0000002A  D
UCB$W_BCR       00000096  D     VCB$L_FCBFL     00000000  D     WCB$W_PREVCOUNT  FFFFFFFA  D
UCB$W_BOFF      0000006C  D     VCB$L_FREE      00000040  D     WCB$W_REFCNT     0000000E  D
UCB$W_BUFQUO    00000018  D     VCB$L_HOME2LBN  00000028  D     WCB$W_SIZE       00000008  D
UCB$W_BYTESTOGO 0000003E  D     VCB$L_HOMELBN   00000024  D
UCB$W_CHARGE    0000004A  D     VCB$L_IBMAPLBN  0000C030  D
UCB$W_CYLINDERS 0000003E  D     VCB$L_IXHDR2LBN 0000002C  D
UCB$W_DA        0000008C  D     VCB$L_MAXFILES  00000044  D
UCB$W_DC        0000008E  D     VCB$L_MVL       00000034  D
UCB$W_DEVBUFSIZ 0000003A  D     VCB$L_QUOCACHE  0000005C  D
UCB$W_DEVSTS    0000005A  D     VCB$L_QUOTAFCB  00000054  D
UCB$W_DIRSEQ    00000088  D     VCB$L_RVT       00000020  D
UCB$W_DSTADDR   00000018  D     VCB$L_SBMAPLBN  00000034  D
UCB$W_DX_BCR    000000A4  D     VCB$L_START_FID 00000028  D
UCB$W_ECT       00000090  D     VCB$L_ST_RECORD 00000030  D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
SEP	000000FB (251.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	FFFFFFFC (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACP$ACCESS	00000000-R	118 (1)	
ACP$ACCESSNET	00000000-R	120 (1)	
ACP$DEACCESS	0000000B-R	163 (1)	
ACP$MODIFY	00000016-R	217 (1)	
ACP$MOUNT	00000021-R	249 (1)	
ACP$READBLK	0000002C-R	289 (1)	
ACP$WRITEBLK	00000049-R	302 (1)	
DEV$V_RCK	=0000001E		#-292 (1)
DEV$V_SQD	=00000005		#-310 (1) #-390 (1)
DEV$V_SWL	=00000019		#-303 (1)
DEV$V_WCK	=0000001F		#-306 (1)
EXE$ABORTIO	00000000-XR		120.2 (1) 163.2 (1) 217.2 (1) 249.2 (1) 353.1 (1)
EXE$FINISHIOC	00000000-XR		299.1 (1) 408.1 (1)
EXE$QIODRVPKT	00000000-XR		400.1 (1)
EXE$READLOCK	00000000-XR		290 (1)
EXE$WRITELOCK	00000000-XR		304 (1)
IO$M_DATACHECK	=00004000		#-293 (1)
IO$M_INHERLOG	=00000800		#-399 (1)
IO$M_INHSEEK	=00001000		#-399 (1)
IO$_LOGICAL	=0000002F		#-319 (1)
IO$_PHYSICAL	=0000001F		#-386 (1)
IO$_WRITELBLK	=00000020		#-388 (1)
IO$_WRITEPBLK	=0000000B		#-388 (1)
IOC$CVTLOGPHY	0000000C-XR		#-398 (1)
IRP$L_MEDIA	0000003C		#-378 (1)
IRP$S_FCODE	=0000000C		#-318 (1) #-385 (1)
IRP$V_FCODE	=00000000		#-318 (1) #-385 (1)
IRP$W_ABCNT	00000030		#-317 (1)
IRP$W_BCNT	00000032		#-315 (1)
IRP$W_FUNC	00000020		#-273 (1) 319 (1) 386 (1) #-389 (1) #-400 (1)
IRP$W_OBCNT	0000003E		#-316 (1)
MXDESCR	=00000016	85 (1)	
P1	=00000000	74 (1)	#-307 (1)
P2	=00000004	75 (1)	#-308 (1)
P3	=00000008	76 (1)	#-314 (1)
P4	=0000000C	77 (1)	
P5	=00000010	78 (1)	
P6	=00000014	79 (1)	
PCB$L_ARB	00000084		#-380 (1)
SS$_BADPARAM	00000000-XR		#-339 (1)
SS$_ENDOFFILE	00000000-XR		#-408 (1)
SS$_FILNOTACC	00000000-XR		#-120.1 (1) #-163.1 (1) #-217.1 (1) #-249.1 (1) #-346 (1)
SS$_ILLBLKNUM	00000000-XR		#-406 (1)
SS$_NOPRIV	00000000-XR		#-353 (1)
SS$_NORMAL	00000000-XR		#-299 (1)
UC[illegible]_DEVCHAR	00000034		292 (1) 303 (1) 306 (1) 310 (1) 390 (1)
UC[illegible]_MAXBLOCK	00000084		#-395 (1)
UCB$L_OWNUIC	0000001C		#-382 (1)
UCB$W_VPROT	0000001A		#-381 (1)
VA$S_BYTE	=00000009		#-392 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CCBDEF	1	53 (1)	53 (1)
$DEFINI	1	53 (1)	53 (1) 54 (1) 55 (1) 56 (1) 57 (1) 58 (1) 59 (1) 60 (1) 61 (1) 62 (1) 63 (1) 64 (1) 65 (1) 66 (1)
$DEVDEF	1	54 (1)	54 (1)
$DYNDEF	2	55 (1)	55 (1)
$IODEF	17	57 (1)	57 (1)
$IPLDEF	1	56 (1)	56 (1)
$IRPDEF	4	58 (1)	58 (1)
$JIBDEF	3	59 (1)	59 (1)
$PCBDEF	4	60 (1)	60 (1)
$PRDEF	4	61 (1)	61 (1)
$PRVDEF	4	62 (1)	62 (1)
$UCBDEF	10	63 (1)	63 (1)
$VADEF	1	64 (1)	64 (1)
$VCBDEF	6	65 (1)	65 (1)
$WCBDEF	3	66 (1)	66 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.15	00:00:00.33
Command processing	123	00:00:00.76	00:00:02.09
Pass 1	677	00:00:18.44	00:00:29.17
Symbol table sort	0	00:00:01.65	00:00:01.77
Pass 2	117	00:00:03.53	00:00:06.21
Symbol table output	30	00:00:00.22	00:00:00.75
Psect synopsis output	6	00:00:00.02	00:00:00.02
Cross-reference output	21	00:00:00.24	00:00:00.25
Assembler run totals	1015	00:00:25.01	00:00:40.61

The working set limit was 1000 pages.
83787 bytes (164 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1185 non-local and 15 local symbols.
333 source lines were read in Pass 1, producing 0 object records in Pass 2.
104 pages of virtual memory were used to define 23 macros.

+--------------------------+
! Macro library statistics !
+--------------------------+

Macro y name	Macros defined
DMA1:[.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	9
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	2
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	19

1568 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ACPFDT/UPDA=(ACPFDT.UPD,ACPFDT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

```
;    0001  0     %title '*** ANSI magtape RMS support'
;    0002  0     module ansi (                          ! Routines to handle Magtape
;    0003  0                  ident = '07-07'
;    0004  0                  ) =
;    0005  0     !
;    0006  0     ! COPYRIGHT (c) 1980, 1981, 1984
;    0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0008  0     !
;    0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0014  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0015  0     ! REMAIN IN DEC.
;    0016  0     !
;    0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0018  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0019  0     ! CORPORATION.
;    0020  0     !
;    0021  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0023  0     !
;    0024  0     !++
;    0025  0     ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0026  0     !
;    0027  0     ! ABSTRACT:
;    0028  0     !
;    0029  0     !       These modules simulate the function of the respective
;    0030  0     !       RMS functions.
;    0031  0     !
;    0032  0     ! AUTHOR:       Roger Riggs, CREATION DATE: 10-November-1979
;    0033  0     !
;    0034  0     ! MODIFIED BY:
;    0035  0     !
;    0036  0     !               Dave Butenhof, 07-oct-1980, version 6.1
;    0037  0     !       01      To allow support of magtape directory command which will
;    0038  0     !               function in standalone or user mode, support minimal level
;    0039  0     !               of wildcarding.  "*" means open next file on tape, return
;    0040  0     !               full file name in NAM block if any.  There is no wildcard
;    0041  0     !               context of any sort; the caller must remember the first
;    0042  0     !               file it found, and exit the wildcard loop when that name
;    0043  0     !               is repeated.  This is for Supervisor use only and will not
;    0044  0     !               be documented outside the program listings of ANSI and
;    0045  0     !               DIRECTORY.
;    0046  0     !       02      Support version numbers
;    0047  0     !               Roger Riggs, 22-oct-1980
;    0048  0     !       03      Added code to implement FAB$V_RWO rewind on open.
;    0049  0     !               The magtape is rewound before the search for the file begins.
;    0050  0     !               also added to code to issue IO$_drvclr before first operation.
;    0051  0     !              -Dave Butenhof, 02-Jun-1981, version 6.4
;    0052  0     !       04      Fix library specifications, use $DS, $DIAG logical names for
;    0053  0     !               flexibility.
;    0054  0     !
;    0055  0     !       05      - Jack Stansbury, 28-Oct-1981, Version 6.-
;    0056  0     !               Changed the PSECT declarations to take out the SEP PSECT.
;    0057  0     !
```

```
;    0058  0      !       06      Bob Bergazzi    May 14, 1984    Version 7.0
;    0059  0      !               Fixed the compiler warning from BLISS released with VMS 4.0.
;    0060  0      !               This occurred at the redeclaration of BOO$QIO with the CALL_BOO
;    0061  0      !               linkage in routine ANSI$OPEN.
;    0062  0      !
;    0063  0      !       07      Bob Bergazzi    May 25, 1984    Version 7.0
;    0064  0      !               Added support for VMS V4 long filenames using HDR4 labels.
;    0065  0      !-
;    0066  0
;    0067  1      begin
;    0068  1      !
;    0069  1      ! include files:
;    0070  1      !
;    0071  1
;    0072  1      library '$diag';                !
;    0073  1
;    0074  1      library '$ds';                  !
;    0075  1
;    0076  1      library 'sys$library:lib';
;    0077  1
;    0078  1      linkage
;    0079  1          jsb = jsb,
;    0080  1          jsb_scan = jsb (register = 3, register = 4, register = 5) : nopreserve (2),
;    0081  1          get_block = call ( ;register = 2) : global (cblock = 9, fab = 11);  !
;    0082  1
;    0083  1      !
;    0084  1      ! table of contents:
;    0085  1      !
;    0086  1
;    0087  1      forward routine
;    0088  1          ansi$open : call_rms addressing_mode (long_relative),       !Open file on magtape
;    0089  1          ansi$close : call_rms addressing_mode (long_relative),      !Close file on magtape
;    0090  1          ansi$read : call_rms addressing_mode (long_relative),       !Read block(s) from magtape
;    0091  1          ansi$get : call_rms addressing_mode (long_relative),        !Get record from magtape
;    0092  1          get_block_length : get_block addressing_mode (long_relative);!Do a BOO$QIO, returning block length
;    0093  1
;    0094  1      !
;    0095  1      ! macros
;    0096  1      !
;    0097  1
;    0098  1
;    0099  1      !
;    0100  1      ! external references:
;    0101  1      !
;    0102  1
;    0103  1      external routine
;    0104  1          scan$numeric : jsb_scan addressing_mode (long_relative),
;    0105  1          kb_check : jsb addressing_mode (long_relative),
;    0106  1          rms$alloc_cache : jsb_cblock addressing_mode (long_relative);
;    0107  1
;    0108  1      !
;    0109  1      ! psect definitions:
;    0110  1      !
;    0111  1
;    0112  1      PSECT                                                           ! [05]
;    0113  1          Plit   = Data (NoExecute, Share, NoWrite,                   ! [05]
;    0114  1                         Addressing_Mode (Long_Relative));            ! [05]
```

```
;    0115  1
;    0116  1     PSECT                                                                    [05]
;    0117  1         Code    = Code (Execute, Share, NoWrite,                             [05]
;    0118  1                         Addressing_Mode (Long_Relative));                    [05]
;    0119  1
;    0120  1     PSECT                                                                    [05]
;    0121  1         Own     = Work (NoExecute, NoShare, Write,                           [05]
;    0122  1                         Addressing_Mode (Long_Relative));                    [05]
;    0123  1
;    0124  1     PSECT                                                                    [05]
;    0125  1         Global = Work (NoExecute, NoShare, Write,                            [05]
;    0126  1                         Addressing_Mode (Long_Relative));                    [05]
```

```
;    0127  1     %sbttl 'ansi$open'
;    0128  1
;    0129  1     global routine ansi$open : call_rms =
;    0130  1
;    0131  1     !++
;    0132  1     ! Functional description:
;    0133  1     !
;    0134  1     !       This routine is responsible for locating the file specified
;    0135  1     !       by the FAB on the magtape specified.  It will search forward
;    0136  1     !       from the current tape position to locate the file.
;    0137  1     !       When a HDR1 record is found the file name is checked,
;    0138  1     !       if the filename matches the rest of the header are skiped.
;    0139  1     !       When a EOF1 record is found the file name is checked,
;    0140  1     !       if the filename matches the tape is backspaced to the start of the data
;    0141  1     !       If at the beginning of the wrong file the entire file is skiped.
;    0142  1     !       If the incorrect file is the same as the first file found when the
;    0143  1     !       search began RMS$_FNF is returned.
;    0144  1     !
;    0145  1     !       If double tape marks are found instead of a file header the
;    0146  1     !       tape is rewound and the search continued until end of tape is
;    0147  1     !       encountered again, when file not found is returned.
;    0148  1     !
;    0149  1     !       For the correct file the entire file is read to determine it length.
;    0150  1     !       The first two blocks read from the tape are saved in the cache
;    0151  1     !       to reduce the amount of positioning for small files.
;    0152  1     !       The current tape position within a file is maintained as TAPEPOS.
;    0153  1     !
;    0154  1     !       If the FAB$V_RWO bit is set in the FAB the tape is rewound before
;    0155  1     !       the search begins
;    0156  1     !
;    0157  1     ! Formal parameters:    none
;    0158  1     !
;    0159  1     ! Implicit inputs:
;    0160  1     !
;    0161  1     !       FAB     Address of related FAB
;    0162  1     !       cblock  Address of per-file context area
;    0163  1     !
;    0164  1     ! Implicit outputs:
;    0165  1     !
;    0166  1     !       FAB     Various fields updated to reflect the file found
;    0167  1     !       cblock  Various fileds updated to reflect the file found
;    0168  1     !
;    0169  1     ! Routine value:
;    0170  1     !
;    0171  1     !       Standard RMS completion codes
;    0172  1     !
;    0173  1     ! Side effects:
;    0174  1     !
;    0175  1     !       The device and associated channel are accessed
;    0176  1     !
;    0177  1     !--
;    0178  1
;    0179  2         BEGIN
;    0180  2
;    0181  2         literal
;    0182  2             true = 1,                                   !Truth
;    0183  2             false = 0,                                  !Falsity
```

```
;    0184  2              end_of_pass1 = 1,                  !Set if end of tape encountered once
;    0185  2              end_of_file = 2;                   !Set if end of file sensed on previous read
;    0186  2
;    0187  2          local
;    0188  2              version,                           ! Version number in file name
;    0189  2              tapeversion,                       ! Version number in hdr1 block
;    0190  2              flag : bitvector [32],             !Contains flag bits
;    0191  2              hdr1 : block [80, byte],           !Contains HDR1 record
;    0192  2              hdr2 : block [80, byte],           !Contains HDR2 record
;    0193  2              hdr4 : block [80, byte],           !Contains HDR3 or HDR4 record                    [07]
;    0194  2              first1 : vector [80, byte],        !Contains first HDR1 record found
;    0195  2              filename : vector [132, byte];     !expected filename in HDR1 label
;    0196  2
;    0197  2          external register
;    0198  2              fab = 11 : ref $fab_decl,          !Base of FAB
;    0199  2              cblock = 9 : ref block [, byte];   !Base of common area
;    0200  2
;    0201  2          external routine                       !                                    [06]
;    0202  2              get_block_length : get_block addressing_mode (long_relative),   !BOO$QIO that returns block_length   [06]
;    0203  2              boo$qio : addressing_mode (long_relative);      !                    [06]
;    0204  2
;    0205  2          bind
;    0206  2              status = fab [fab$l_stv];          !Return value from boo$qio
;    0207  2
;    0208  2      !
;    0209  2      ! Initialize dispatch addresses
;    0210  2      !
;    0211  2          cblock [ctl$l_get] = ansi$get;
;    0212  2          cblock [ctl$l_read] = ansi$read;
;    0213  2          cblock [ctl$l_close] = ansi$close;
;    0214  2      !+
;    0215  2      ! convert the file name in the control region into the format
;    0216  2      ! expected for comparison with HDR1 labels.
;    0217  2      ! note that the version number is ignored.
;    0218  2      !-
;    0219  3          begin
;    0220  3
;    0221  3          local
;    0222  3              len,
;    0223  3              verlen,
;    0224  3              veradr,
;    0225  3              adr;                               !Length and address of current segment
;    0226  3
;    0227  3          adr = ch$find_ch (.cblock [ctl$w_file_len], .cblock [ctl$l_file_ptr], %C']') + 1;
;    0228  3          veradr = ch$find_ch (.cblock [ctl$w_file_len], .cblock [ctl$l_file_ptr], %C';');
;    0229  3          len = .veradr - .adr;                  ! Calculate length of <name>.<type>
;    0230  3          veradr = .veradr + 1;                  ! Address of version (after ";")
;    0231  3          verlen = .cblock [ctl$w_file_len] - (.veradr - .cblock [ctl$l_file_ptr]);   ! Length of rest of name
;    0232  3          version = scan$numeric (10, .verlen, .veradr);      ! Convert ascii to binary
;    0233  3          ch$fill (' ', 132, filename);
;    0234  3          ch$move (.len, .adr, filename);
;    0235  2          end;
;    0236  2          flag = false;
;    0237  2          first1 [0] = 0;                        !Clear first character
;    0238  2
;    0239  3          if not (status = boo$qio (0, 0, 0, io$_drvclr, 0, .cblock [ctl$l_rpb]))
;    0240  2          then return .status;
```

```
;    0241  2
;    0242  2          if .fab [fab$v_rwo]
;    0243  2          then
;    0244  3              begin
;    0245  4              if not (status = boo$qio (0, 0, 0, io$_rewind, 0, .cblock [ctl$l_rpb]))
;    0246  3              then return .status;
;    0247  2              end;
;    0248  2
;    0249  2          until 0 do
;    0250  3              begin
;    0251  3              kb_check ();                                !Check for ^C
;    0252  3              status = boo$qio (hdr1, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);
;    0253  3
;    0254  3              selectone .status of
;    0255  3                  set
;    0256  3
;    0257  3                  [ss$_normal] :
;    0258  4                      begin
;    0259  4                      flag [end_of_file] = false;
;    0260  4
;    0261  4                      if ((.hdr1 eql %ascii'EOF1') or (.hdr1 eql %ascii'HDR1')) and
;    0262  5                          begin
;    0263  5                          tapeversion = (scan$numeric (10, 4, hdr1 [hd1$t_genno]) - 1)*100 + scan$numeric (10, 2,
;    0264  5                              hdr1 [hd1$t_genver]) + 1;       ! Calculate version number
;    0265  7                          if ((ch$compare (17, filename, 17, hdr1 [hd1$t_fileid], 0) eql 0) or (.filename [0] eql %c
;    0266  6                              '*' and .hdr1 eql %ascii'HDR1')) and (.tapeversion eql .version or .version eql 0)
;    0267  6                              then begin
;    0268  6                                   status = boo$qio (hdr2, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);
;    0269  6                                   if not .status then return rms$_rer;
;    0270  6
;    0271  6                              status = boo$qio (hdr4, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);      ! Really
;    0272  6                              if not .status then return rms$_rer;                                       ! reads HDR3
;    0273  6
;    0274  6                              status = boo$qio (hdr4, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);      ! Reads HDR4
;    0275  6
;    0276  7                              if ((.filename [0] eql %c'*') and (.version eql 0))                        ! If wildcar
;    0277  6                                 then true                                                               !
;    0278  6                                 else if .status eql ss$_endoffile                                       ! EOF means
;    0279  6                                         then true                                                       ! no HDR4
;    0280  6                                         else if not .status                                             ! Error whic
;    0281  6                                                 then return rms$_rer                                    ! is not EOF
;    0282  6                                                 else if .hdr4 [hd4$b_fileid_ext_size] eql 0     ! Filename fits in H
;    0283  6                                                         then true                                       !
;    0284  6                                                         else            ! Compare the rest of the filename in HDR4
;    0285  6                                                              if ch$compare (.hdr4 [hd4$b_fileid_ext_size],      !
;    0286  6                                                                             filename [17],                      !
;    0287  6                                                                             .hdr4 [hd4$b_fileid_ext_size],      !
;    0288  6                                                                             hdr4 [hd4$t_fileid_ext], 0) eql 0 !
;    0289  6                                                                 then true                                      !
;    0290  6                                                                 else false                                     !
;    0291  6                          end
;    0292  5                      else false
;    0293  5                  end
;    0294  4          then
;    0295  5              begin
;    0296  5
;    0297  5              local
```

```
;    0298  5                              block_length;
;    0299  5
;    0300  5      !
;    0301  5      ! KLUGE:
;    0302  5      !       This module does NOT actually support $NAM block.  However, if one
;    0303  5      !       is present, it will copy the ''resultant name string'' into the RSA
;    0304  5      !       field.  This supports the DS ''Directory'' command and has no other
;    0305  5      !       purpose.
;    0306  5      !
;    0307  5
;    0308  5                          if .fab [fab$l_nam] neq 0
;    0309  5                          then
;    0310  6                              begin
;    0311  6
;    0312  6                              bind
;    0313  6                                  phynam = .cblock [ctl$l_ptable] : bblock,
;    0314  6                                  nam = .fab [fab$l_nam] : bblock;
;    0315  6
;    0316  6                              local
;    0317  6                                  blank,
;    0318  6                                  dev : bblock [dsc$c_s_bln],
;    0319  6                                  file : bblock [dsc$c_s_bln],
;    0320  6                                  longfile : bblock [dsc$c_s_bln],                    !       [07]
;    0321  6                                  a : bblock [dsc$c_s_bln];
;    0322  6
;    0323  6                              a [dsc$w_length] = .nam [nam$b_rss];
;    0324  6                              a [dsc$a_pointer] = .nam [nam$l_rsa];
;    0325  6                              dev [dsc$a_pointer] = .phynam [dsc$a_pointer] + 1;      ! Trim off ''_''
;    0326  6                              dev [dsc$w_length] = .phynam [dsc$w_length] - 1;
;    0327  6                              file [dsc$a_pointer] = hdr1 [hd1$t_fileid];
;    0328  6                              blank = ch$find_ch (17, hdr1 [hd1$t_fileid], %c' ');
;    0329  6                              file [dsc$w_length] = (if .blank eql 0 then 17 else .blank - hdr1 [hd1$t_fileid]);
;    0330  6                              longfile [dsc$a_pointer] = hdr4 [hd4$t_fileid_ext];     !       [07]
;    0331  7                              if ((.hdr4 eql %ascii'HDR4') or (.hdr4 eql %ascii'EOF4'))   !       [07]
;    0332  6                                 then longfile [dsc$w_length] = .hdr4 [hd4$b_fileid_ext_size]   !       [07]
;    0333  6                                 else longfile [dsc$w_length] = 0;                   !       [07]
;    0334  6                              $fao ($ascid (''!AS:!AS!AS;!ZL''), a, a, dev, file, longfile, .tapeversion);   !       [07]
;    0335  6                                                  ! Convert version to ascii
;    0336  6                              nam [nam$b_rsl] = .a [dsc$w_length]
;    0337  5                              end;
;    0338  5
;    0339  5                          if .hdr1 eql %ascii'HDR1'
;    0340  6                          then begin                                              !       [07]
;    0341  6                               if .hdr4 eql %ascii'HDR4'      ! At start of file, skip leading EOF    [07]
;    0342  7                                  then begin                  ! Already there if pre-VMS V4 tape      [07]
;    0343  7                                       status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0344  7                                       if not .status then return rms$_rer
;    0345  7                                       end
;    0346  6                               end
;    0347  6                          else begin                          ! at end of file, backover trailers and data
;    0348  6                               if .hdr4 neq %ascii'EOF4'      ! Backup an extra tape mark             [07]
;    0349  7                                  then begin                                           !       [07]
;    0350  7                                       status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);   !       [07]
;    0351  7                                       if not .status then return rms$_rer          !       [07]
;    0352  6                                       end;                                         !       [07]
;    0353  6
;    0354  6                              status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
```

```
;    0355  6                        if not .status then return rms$_rer;
;    0356  6
;    0357  6                        status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0358  6
;    0359  6                        if not .status then return rms$_rer;
;    0360  6
;    0361  6                        status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0362  6
;    0363  6                        if not .status then return rms$_rer;
;    0364  6
;    0365  6                        end;
;    0366  5
;    0367  5                    if .(hdr2 [hd2$t_recatr1]) neq %ascii'    '
;    0368  5                    then
;    0369  6                        begin
;    0370  6
;    0371  6                        bind
;    0372  6                            fat = hdr2 [hd2$t_recatr1] : block [fat$c_length, byte];
;    0373  6
;    0374  6                        fab [fab$b_rat] = .fat [fat$b_rattrib];
;    0375  6                        fab [fab$b_rfm] = .fat [fat$v_rtype];
;    0376  6                        fab [fab$b_fsz] = .fat [fat$b_vfcsize];
;    0377  6                        fab [fab$w_mrs] = .fat [fat$w_maxrec];
;    0378  6                        end
;    0379  6                    else
;    0380  5                        begin
;    0381  6
;    0382  6                        selectone .hdr2 [hd2$b_recformat] of
;    0383  6                            set
;    0384  6
;    0385  6                            [%c'F'] :
;    0386  6                                begin
;    0387  7                                fab [fab$b_rfm] = fab$c_fix;
;    0388  7                                fab [fab$w_mrs] = scan$numeric (10, 5, hdr2 [hd2$t_reclen]);
;    0389  7                                end;
;    0390  6
;    0391  6                            [%c'D'] :
;    0392  6                                begin
;    0393  7                                fab [fab$b_rfm] = fab$c_var;
;    0394  7                                fab [fab$w_mrs] = scan$numeric (10, 5, hdr2 [hd2$t_reclen]) - 4;
;    0395  7                                end;
;    0396  6
;    0397  6                            [otherwise] :
;    0398  6                                return rms$_org;
;    0399  6                            tes;
;    0400  6
;    0401  6                        selectone .hdr2 [hd2$b_formcntrl] of
;    0402  6                            set
;    0403  6
;    0404  6                            [%c'A'] :
;    0405  6                                fab [fab$b_rat] = fab$m_ftn;
;    0406  6
;    0407  6                            [%c' '] :
;    0408  6                                fab [fab$b_rat] = fab$m_cr;
;    0409  6
;    0410  6                            [otherwise] :
;    0411  6
```

```
;    0412  6                                    fab [fab$b_rat] = 0;
;    0413  6                                tes;
;    0414  6
;    0415  5                                end;
;    0416  5
;    0417  6                            if not (status = rms$alloc_cache ((scan$numeric (10, 5, hdr2 [hd2$t_blocklen]) + 511)/512
;    0418  6                                ))
;    0419  5                            then
;    0420  5                                return rms$_dme;
;    0421  5
;    0422  5                            cblock [ctl$l_eofvbn] = 0;
;    0423  5                            cblock [ctl$w_offbyte] = 0;
;    0424  5                            cblock [ctl$l_lvbn] = cblock [ctl$l_hvbn] = 1;
;    0425  5                            cblock [ctl$l_tapepos] = 1;
;    0426  5
;    0427  5                            while get_block_length( ;block_length) do           !                                   [06]
;    0428  6                                begin
;    0429  6
;    0430  6                                bind
;    0431  6                                    size = cblock [ctl$w_cache_size] : word;
;    0432  6
;    0433  6                                if .cblock [ctl$l_tapepos] eql 1
;    0434  6                                then
;    0435  7                                    begin
;    0436  7
;    0437  7                                    local
;    0438  7                                        tmp;
;    0439  7
;    0440  7                                    tmp = .cblock [ctl$l_cache3];
;    0441  7                                    cblock [ctl$l_cache3] = .cblock [ctl$l_cache1];
;    0442  7                                    cblock [ctl$l_cache1] = .tmp;
;    0443  7                                    cblock [ctl$l_hvbn] = .size + 1;
;    0444  6                                    end;
;    0445  6
;    0446  6                                if .cblock [ctl$l_tapepos] eql .size + 1
;    0447  6                                then
;    0448  7                                    begin
;    0449  7
;    0450  7                                    local
;    0451  7                                        tmp;
;    0452  7
;    0453  7                                    tmp = .cblock [ctl$l_cache3];
;    0454  7                                    cblock [ctl$l_cache3] = .cblock [ctl$l_cache2];
;    0455  7                                    cblock [ctl$l_cache2] = .tmp;
;    0456  7                                    cblock [ctl$l_hvbn] = .size^1 + 1;
;    0457  6                                    end;
;    0458  6
;    0459  6                                cblock [ctl$l_tapepos] = .cblock [ctl$l_tapepos] + .size;
;    0460  6                                cblock [ctl$l_eofvbn] = ((.cblock [ctl$l_eofvbn] + .size - 1) and not (.size - 1)) +
;    0461  6                                ((.block_length + 511)^-9);
;    0462  6                                cblock [ctl$w_offbyte] = .block_length and 511;
;    0463  5                                end;
;    0464  5
;    0465  5                            cblock [ctl$l_eofvbn] = .cblock [ctl$l_eofvbn] + 1;
;    0466  5                            status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0467  5
;    0468  5                            if not .status then return rms$_rer;
```

```
;    0469  5
;    0470  5                        Return RMS$_normal
;    0471  5                        end
;    0472  4                    else
;    0473  5                        begin                        ! Not correct file
;    0474  5
;    0475  5                        if .hdr1 eql 'HDR1'          ! If at start of file skip whole file
;    0476  5                        then
;    0477  6                            begin
;    0478  6
;    0479  6                            if ch$compare (80, hdr1, 80, first1, 0) eql 0
;    0480  6                            then
;    0481  7                                begin
;    0482  7                                if .hdr4 eql %ascii'HDR4'                                          !       [07]
;    0483  8                                    then begin                                                     !       [07]
;    0484  8                                         status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]); !  [07]
;    0485  8                                         if not .status then return rms$_rer;                      !       [07]
;    0486  8                                         end                                                       !       [07]
;    0487  8                                    else begin                                 ! Must be pre-VMS V4 tape (no HDR4)  [07]
;    0488  8                                         status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]); !    [07]
;    0489  8                                         if not .status then return rms$_rer;                      !       [07]
;    0490  7                                         end;                                                      !       [07]
;    0491  7                                status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);   !       [07]
;    0492  7                                if not .status then return rms$_rer;                               !       [07]
;    0493  7                                status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);   !       [07]
;    0494  7                                if not .status then return rms$_rer;                               !       [07]
;    0495  7                                status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);
;    0496  7                                if not .status then return rms$_rer;
;    0497  7                                return rms$_fnf;
;    0498  7                                end
;    0499  6                            else
;    0500  6
;    0501  6                                if .first1 [0] eql 0 then ch$move (80, hdr1, first1);
;    0502  6
;    0503  6                            if .hdr4 eql %ascii'HDR4'                                              !       [07]
;    0504  7                               then begin                                                          !       [07]
;    0505  7                                    status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0506  7                                    if not .status then return rms$_rer;
;    0507  6                                    end;
;    0508  6
;    0509  6                            status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0510  6
;    0511  6                            if not .status then return rms$_rer;
;    0512  6
;    0513  6                            status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0514  6
;    0515  6                            if not .status then return rms$_rer;
;    0516  6
;    0517  6                            flag [end_of_file] = true;
;    0518  5                            end;
;    0519  5
;    0520  4                        end;
;    0521  4
;    0522  3                    end;
;    0523  3
;    0524  3                [ss$_dataoverun] :
;    0525  4                    begin
```

```
;    0526  4                    status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0527  4
;    0528  4                    if not .status then return rms$_rer;
;    0529  4
;    0530  4                    flag [end_of_file] = false;
;    0531  3                    end;
;    0532  3
;    0533  3                [ss$_endoffile] :
;    0534  4                    begin
;    0535  4
;    0536  4                    if .flag [end_of_file]
;    0537  4                    then
;    0538  5                        begin
;    0539  5                        status = boo$qio (0, 0, 0, io$_rewind, 0, .cblock [ctl$l_rpb]);
;    0540  5                        flag [end_of_file] = false;
;    0541  5
;    0542  5                        if .flag [end_of_pass1] then return rms$_fnf else flag [end_of_pass1] = true;
;    0543  5
;    0544  5                        end
;    0545  4                    else
;    0546  4                        flag [end_of_file] = true;
;    0547  4
;    0548  3                    end;
;    0549  3
;    0550  3                [otherwise] :
;    0551  3                    return rms$_rer;
;    0552  3                tes;
;    0553  3
;    0554  2            end;
;    0555  2
;    0556  2            ss$_normal
;    0557  1            END;


                                                                .TITLE  ANSI *** ANSI magtape RMS support
                                                                .IDENT  \07-07\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

    4C  5A  21  3B  53  41  21  53  41  21  3A  53  41  21  00000 P.AAB:  .ASCII  \!AS:!AS!AS;!ZL\                    ;
                                                            0000E       .BLKB   2
                                                  0000000E  00010 P.AAA:  .LONG   14                                 ;
                                                  00000000' 00014        .ADDRESS P.AAB                             ;

                                                                .EXTRN  SCAN$NUMERIC, KB_CHECK
                                                                .EXTRN  RMS$ALLOC_CACHE
                                                                .EXTRN  GET_BLOCK_LENGTH
                                                                .EXTRN  BOO$QIO, SYS$FAO

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                      05FC 00000        .ENTRY  ANSI$OPEN, Save R2,R3,R4,R5,R6,R7,R8,R10    ; 0129
                                          5E      FE14   CE   9E 00002  MOVAB   -492(SP), SP                            ;
                                          56        0C   AB   9E 00007  MOVAB   12(FAB), R6                            ; 0206
                                      0C  A9 00000000V   EF   9E 0000B  MOVAB   ANSI$GET, 12(CBLOCK)                   ; 0211
                                      10  A9 00000000V   EF   9E 00013  MOVAB   ANSI$READ, 16(CBLOCK)                  ; 0212
                                      14  A9 00000000V   EF   9E 0001B  MOVAB   ANSI$CLOSE, 20(CBLOCK)                 ; 0213
```

```
                4C   B9        48   A9        5D   8F   3A 00023          LOCC    #93, 72(CBLOCK), @76(CBLOCK)                ; 0227
                                                  02   12 0002A          BNEQ    1$                                          ;
                                                  51   D4 0002C          CLRL    R1                                          ;
                                        57   01   A1   9E 0002E 1$:       MOVAB   1(R1), ADR                                  ;
                4C   B9        48   A9           3B   3A 00032          LOCC    #59, 72(CBLOCK), @76(CBLOCK)                ; 0228
                                                  02   12 00038          BNEQ    2$                                          ;
                                                  51   D4 0003A          CLRL    R1                                          ;
                     58             51           57   C3 0003C 2$:       SUBL3   ADR, VERADR, LEN                            ; 0229
                                                  51   D6 00040          INCL    VERADR                                      ; 0230
                     50        4C   A9           51   C3 00042          SUBL3   VERADR, 76(CBLOCK), R0                      ; 0231
                                   54   48   A9   3C   00047          MOVZWL  72(CBLOCK), VERLEN                          ;
                                   54           50   C0 0004B          ADDL2   R0, VERLEN                                  ;
                                   55           51   D0 0004E          MOVL    VERADR, R5                                  ; 0232
                                   53           0A   D0 00051          MOVL    #10, R3                                     ;
                                        00000000G EF   16 00054          JSB     SCAN$NUMERIC                                ;
                                   6E           50   D0 0005A          MOVL    R0, VERSION                                 ;
    0084   8F        20             6E           00   2C 0005D          MOVC5   #0, (SP), #32, #132, FILENAME               ; 0233
                                             28   AE      00064                                                              ;
                4C   AE             67           58   28 00066          MOVC3   LEN, (ADR), FILENAME                        ; 0234
                                                  5A   D4 0006B          CLRL    FLAG                                        ; 0236
                                   00AC          CE   94 0006D          CLRB    FIRST1                                      ; 0237
                                        38       DD   00071             PUSHL   56(CBLOCK)                                  ; 0239
                                   7E            04   7D 00074          MOVQ    #4, -(SP)                                   ;
                                                 7E   7C 00077          CLRQ    -(SP)                                       ;
                                                 7E   D4 00079          CLRL    -(SP)                                       ;
                          00000000G EF           06   FB 0007B          CALLS   #6, BOO$QIO                                 ;
                                   66           50   D0 00082          MOVL    R0, (R6)                                    ;
                                   1C           50   E9 00085          BLBC    R0, 3$                                      ;
                                             04   AB   95 00088          TSTB    4(FAB)                                      ; 0242
                                             1B   18 0008B              BGEQ    4$                                          ;
                                        38   A9   DD   0008D             PUSHL   56(CBLOCK)                                  ; 0245
                                   7E            24   7D 00090          MOVQ    #36, -(SP)                                  ;
                                                 7E   7C 00093          CLRQ    -(SP)                                       ;
                                                 7E   D4 00095          CLRL    -(SP)                                       ;
    00000000G                      EF           06   FB 00097          CALLS   #6, BOO$QIO                                 ;
                                   66           50   D0 0009E          MOVL    R0, (R6)                                    ;
                                   04           50   E8 000A1          BLBS    R0, 4$                                      ;
                                   50           66   D0 000A4 3$:       MOVL    (R6), R0                                    ; 0246
                                                04   000A7             RET                                                 ;
                          00000000G EF           16   000A8 4$:       JSB     KB_CHECK                                    ; 0251
                                   58        38   A9   D0 000AE          MOVL    56(CBLOCK), R8                              ; 0252
                                             58   DD   000B2             PUSHL   R8                                          ;
                                   7E            21   7D 000B4          MOVQ    #33, -(SP)                                  ;
                                                 7E   D4 000B7          CLRL    -(SP)                                       ;
                                   7E        50   8F   9A 000B9          MOVZBL  #80, -(SP)                                  ;
                                             B0   AD   9F 000BD          PUSHAB  HDR1                                        ;
    00000000G                      EF           06   FB 000C0          CALLS   #6, BOO$QIO                                 ;
                                   66           50   D0 000C7          MOVL    R0, (R6)                                    ;
                                   01           66   D1 000CA          CMPL    (R6), #1                                    ; 0257
                                                03   13 000CD          BEQL    5$                                          ;
                                              0470   31 000CF          BRW     54$                                         ;
                                   5A           04   8A 000D2 5$:       BICB2   #4, FLAG                                    ; 0259
    31464F45                       8F        B0   AD   D1 000D5          CMPL    HDR1, #826691397                            ; 0261
                                             0A   13 000DD              BEQL    6$                                          ;
    31524448                       8F        B0   AD   D1 000DF          CMPL    HDR1, #827475016                            ;
                                             5A   12 000E7              BNEQ    9$                                          ;
                                   55        D3   AD   9E 000E9 6$:       MOVAB   HDR1+35, R5                                 ; 0263
```

```
                                54              04   D0 000ED          MOVL    #4, R4                                          ;
                                53              0A   D0 000F0          MOVL    #10, R3                                         ;
                                   00000000G    EF   16 000F3          JSB     SCAN$NUMERIC                                    ;
                                57              50   D0 000F9          MOVL    R0, R7                                          ;
                                57 00000064     8F   C4 000FC          MULL2   #100, R7                                        ;
                                55        D7    AD   9E 00103          MOVAB   HDR1+39, R5                                     ; 0264
                                54              02   D0 00107          MOVL    #2, R4                                          ;
                                53              0A   D0 0010A          MOVL    #10, R3                                         ;
                                   00000000G    EF   16 0010D          JSB     SCAN$NUMERIC                                    ;
                          04    AE        9D A047   9E 00113          MOVAB   -99(R0)[R7], TAPEVERSION                        ;
                                54              01   D0 00119          MOVL    #1, R4                                          ; 0265
                    B4  AD        28    AE        11   29 0011C          CMPC3   #17, FILENAME, HDR1+4                           ;
                                              03   1A 00122          BGTRU   7$                                              ;
                                54              01   D9 00124          SBWC    #1, R4                                          ;
                                              54   D5 00127 7$:       TSTL    R4                                              ;
                                              10   13 00129          BEQL    8$                                              ;
                                2A        28    AE   91 0012B          CMPB    FILENAME, #42                                   ;
                                              12   12 0012F          BNEQ    9$                                              ;
                          31524448  8F        B0   AD   D1 00131          CMPL    HDR1, #827475016                        ; 0266
                                              08   12 00139          BNEQ    9$                                              ;
                                6E        04    AE   D1 0013B 8$:       CMPL    TAPEVERSION, VERSION                            ;
                                              07   13 0013F          BEQL    10$                                             ;
                                              6E   D5 00141          TSTL    VERSION                                         ;
                                              03   13 00143 9$:       BEQL    10$                                             ;
                                           0084   31 00145          BRW     15$                                             ;
                                              58   DD 00148 10$:      PUSHL   R8                                              ; 0268
                                7E              21   7D 0014A          MOVQ    #33, -(SP)                                      ;
                                              7E   D4 0014D          CLRL    -(SP)                                           ;
                                7E        50    8F   9A 0014F          MOVZBL  #80, -(SP)                                      ;
                                       FF60     CD   9F 00153          PUSHAB  HDR2                                            ;
                   00000000G    EF              06   FB 00157          CALLS   #6, BOO$QIO                                     ;
                                66              50   D0 0015E          MOVL    R0, (R6)                                        ;
                                48              66   E9 00161          BLBC    (R6), 12$                                       ; 0269
                                              58   DD 00164          PUSHL   R8                                              ; 0271
                                7E              21   7D 00166          MOVQ    #33, -(SP)                                      ;
                                              7E   D4 00169          CLRL    -(SP)                                           ;
                                7E        50    3F   9A 0016B          MOVZBL  #80, -(SP)                                      ;
                                       FF10     CD   9F 0016F          PUSHAB  HDR4                                            ;
                   00000000G    EF              06   FB 00173          CALLS   #6, BOO$QIO                                     ;
                                66              50   D0 0017A          MOVL    R0, (R6)                                        ;
                                2C              66   E9 0017D          BLBC    (R6), 12$                                       ; 0272
                                              58   DD 00180          PUSHL   R8                                              ; 0274
                                7E              21   7D 00182          MOVQ    #33, -(SP)                                      ;
                                              7E   D4 00185          CLRL    -(SP)                                           ;
                                7E        50    8F   9A 00187          MOVZBL  #80, -(SP)                                      ;
                                       FF10     CD   9F 0018B          PUSHAB  HDR4                                            ;
                   00000000G    EF              06   FB 0018F          CALLS   #6, BOO$QIO                                     ;
                                66              50   D0 00196          MOVL    R0, (R6)                                        ;
                                2A        28    AE   91 00199          CMPB    FILENAME, #42                                   ; 0276
                                              04   12 0019D          BNEQ    11$                                             ;
                                              6E   D5 0019F          TSTL    VERSION                                         ;
                                              2C   13 001A1          BEQL    16$                                             ;
                   00000870     8F              66   D1 001A3 11$:       CMPL    (R6), #2160                                     ; 0278
                                              23   13 001AA          BEQL    16$                                             ;
                                03              66   E8 001AC 12$:       BLBS    (R6), 13$                                       ; 0280
                                           03EE   31 001AF          BRW     61$                                             ;
                                50        FF14    CD   9A 001B2 13$:       MOVZBL  HDR4+4, R0                                      ; 0282
```

```
                                          16   13 001B7         BEQL    16$                              ;
                              54          01   D0 001B9         MOVL    #1, R4                           ; 0288
      FF15    CD       39     AE          50   29 001BC         CMPC3   R0, FILENAME+17, HDR4+5          ;
                                          03   1A 001C3         BGTRU   14$                              ;
                              54          01   D9 001C5         SBWC    #1, R4                           ;
                                          54   D5 001C8 14$:    TSTL    R4                               ;
                                          03   13 001CA         BEQL    16$                              ;
                                        0282   31 001CC 15$:    BRW     42$                              ;
                              52     28   AB   D0 001CF 16$:    MOVL    40(FAB), R2                      ; 0308
                                          03   12 001D3         BNEQ    17$                              ;
                                        0086   31 001D5         BRW     24$                              ;
                              50     3C   A9   D0 001D8 17$:    MOVL    60(CBLOCK), R0                   ; 0313
                     08       AE     02   A2   9B 001DC         MOVZBW  2(R2), A                         ; 0323
                     0C       AE     04   A2   D0 001E1         MOVL    4(R2), A+4                       ; 0324
        24    AE     04       A0          01   C1 001E6         ADDL3   #1, 4(R0), DEV+4                 ; 0325
        20    AE              60          01   A3 001EC         SUBW3   #1, (R0), DEV                    ; 0326
                     1C       AE     B4   AD   9E 001F1         MOVAB   HDR1+4, FILE+4                   ; 0327
        B4    AD              11          20   3A 001F6         LOCC    #32, #17, HDR1+4                 ; 0328
                                          02   12 001FB         BNEQ    18$                              ;
                                          51   D4 001FD         CLRL    R1                               ;
                                          51   D5 001FF 18$:    TSTL    BLANK                            ; 0329
                                          05   12 00201         BNEQ    19$                              ;
                              51          11   D0 00203         MOVL    #17, R1                          ;
                                          07   11 00206         BRB     20$                              ;
                              50     B4   AD   9E 00208 19$:    MOVAB   HDR1+4, R0                       ;
                              51          50   C2 0020C         SUBL2   R0, R1                           ;
                     18       AE          51   B0 0020F 20$:    MOVW    R1, FILE                         ;
                     14       AE   FF15   CD   9E 00213         MOVAB   HDR4+5, LONGFILE+4               ; 0330
              34524448        8F   FF10   CD   D1 00219         CMPL    HDR4, #877806664                 ; 0331
                                          0B   13 00222         BEQL    21$                              ;
              34464F45        8F   FF10   CD   D1 00224         CMPL    HDR4, #877023045                 ;
                                          08   12 0022D         BNEQ    22$                              ;
                     10       AE   FF14   CD   9B 0022F 21$:    MOVZBW  HDR4+4, LONGFILE                 ; 0332
                                          03   11 00235         BRB     23$                              ;
                                     10   AE   B4 00237 22$:    CLRW    LONGFILE                         ; 0333
                                     04   AE   DD 0023A 23$:    PUSHL   TAPEVERSION                      ; 0334
                                     14   AE   9F 0023D         PUSHAB  LONGFILE                         ;
                                     20   AE   9F 00240         PUSHAB  FILE                             ;
                                     2C   AE   9F 00243         PUSHAB  DEV                              ;
                                     18   AE   9F 00246         PUSHAB  A                                ;
                                     1C   AE   9F 00249         PUSHAB  A                                ;
                               00000000'  EF   9F 0024C         PUSHAB  P.AAA                            ;
              00000000G       00          07   FB 00252         CALLS   #7, SYS$FAO                      ;
                    03        A2     08   AE   90 00259         MOVB    A, 3(R2)                         ; 0336
              31524448        8F     B0   AD   D1 0025E 24$:    CMPL    HDR1, #827475016                 ; 0339
                                          24   12 00266         BNEQ    27$                              ;
              34524448        8F   FF10   CD   D1 00268         CMPL    HDR4, #877806664                 ; 0341
                                          6B   12 00271         BNEQ    29$                              ;
                                          58   DD 00273 25$:    PUSHL   R8                               ; 0343
                              7E          25   7D 00275         MOVQ    #37, -(SP)                       ;
                                          7E   7C 00278         CLRQ    -(SP)                            ;
                                          01   DD 0027A         PUSHL   #1                               ;
              00000000G       EF          06   FB 0027C         CALLS   #6, BOO$QIO                      ;
                              66          50   D0 00283         MOVL    R0, (R6)                         ;
                              55          66   E8 00286         BLBS    (R6), 29$                        ; 0344
                                        0314   31 00289 26$:    BRW     61$                              ;
              34464F45        8F   FF10   CD   D1 0028C 27$:    CMPL    HDR4, #877023045                 ; 0348
```

```
                                                17  13 00295          BEQL    28$                                     ; 0350
                                                58  DD 00297          PUSHL   R8
                                      7E        25  7D 00299          MOVQ    #37, -(SP)
                                                7E  7C 0029C          CLRQ    -(SP)
                                      7E        01  CE 0029E          MNEGL   #1, -(SP)
                           00000000G  EF        06  FB 002A1          CALLS   #6, B00$QIO
                                      66        50  D0 002A8          MOVL    R0, (R6)
                                      DB        66  E9 002AB          BLBC    (R6), 26$                               ; 0351
                                                58  DD 002AE 28$:     PUSHL   R8                                      ; 0354
                                      7E        25  7D 002B0          MOVQ    #37, -(SP)
                                                7E  7C 002B3          CLRQ    -(SP)
                                      7E        01  CE 002B5          MNEGL   #1, -(SP)
                           00000000G  EF        06  FB 002B8          CALLS   #6, B00$QIO
                                      66        50  D0 002BF          MOVL    R0, (R6)
                                      C4        66  E9 002C2          BLBC    (R6), 26$                               ; 0356
                                                58  DD 002C5          PUSHL   R8                                      ; 0358
                                      7E        25  7D 002C7          MOVQ    #37, -(SP)
                                                7E  7C 002CA          CLRQ    -(SP)
                                      7E        01  CE 002CC          MNEGL   #1, -(SP)
                           00000000G  EF        06  FB 002CF          CALLS   #6, B00$QIO
                                      66        50  D0 002D6          MOVL    R0, (R6)
                                      AD        66  E9 002D9          BLBC    (R6), 26$                               ; 0360
                                                95  11 002DC          BRB     25$                                     ; 0362
                           20202020   8F    FF6F  CD  D1 002DE 29$:     CMPL    HDR2+15, #538976288                     ; 0368
                                                1F  13 002E7          BEQL    30$
                                 1E    AB    FF70  CD  90 002E9          MOVB    FAT+1, 30(FAB)                          ; 0375
      50    FF6F    CD                    04        00  EF 002EF          EXTZV   #0, #4, FAT, R0                         ; 0376
                                 1F    AB          50  90 002F6          MOVB    R0, 31(FAB)
                                 3F    AB    FF7E  CD  90 002FA          MOVB    FAT+15, 63(FAB)                         ; 0377
                                 36    AB    FF7F  CD  B0 00300          MOVW    FAT+16, 54(FAB)                         ; 0378
                                                6E  11 00306          BRB     36$                                     ; 0368
                                       50   FF64  CD  9A 00308 30$:     MOVZBL  HDR2+4, R0                              ; 0383
                                 46    8F          50  91 0030D          CMPB    R0, #70                                 ; 0386
                                                1B  12 00311          BNEQ    31$
                                 1F    AB          01  90 00313          MOVB    #1, 31(FAB)                             ; 0388
                                       55   FF6A  CD  9E 00317          MOVAB   HDR2+10, R5                             ; 0389
                                       54        05  D0 0031C          MOVL    #5, R4
                                       53        0A  D0 0031F          MOVL    #10, R3
                                            00000000G  EF  16 00322          JSB     SCAN$NUMERIC
                                 36    AB          50  B0 00328          MOVW    R0, 54(FAB)
                                                2A  11 0032C          BRB     33$                                     ; 0383
                                 44    8F          50  91 0032E 31$:     CMPB    R0, #68                                 ; 0392
                                                1C  12 00332          BNEQ    32$
                                 1F    AB          02  90 00334          MOVB    #2, 31(FAB)                             ; 0394
                                       55   FF6A  CD  9E 00338          MOVAB   HDR2+10, R5                             ; 0395
                                       54        05  D0 0033D          MOVL    #5, R4
                                       53        0A  D0 00340          MOVL    #10, R3
                                            00000000G  EF  16 00343          JSB     SCAN$NUMERIC
            36    AB                    50        04  A3 00349          SUBW3   #4, R0, 54(FAB)
                                                08  11 0034E          BRB     33$                                     ; 0383
                                       50 0001860C  8F  D0 00350 32$:     MOVL    #99852, R0                              ; 0399
                                                04 00357          RET
                                       50     84    AD  9A 00358 33$:     MOVZBL  HDR2+36, R0                             ; 0402
                                 41    8F          50  91 0035C          CMPB    R0, #65                                 ; 0405
                                                06  12 00360          BNEQ    34$
                                 1E    AB          01  90 00362          MOVB    #1, 30(FAB)                             ; 0406
                                                0E  11 00366          BRB     36$
```

```
                                20              50  91 00368 34$:    CMPB    R0, #32                                 ; 0408
                                                06  12 0036B        BNEQ    35$                                     ;
                             1E AB              02  90 0036D        MOVB    #2, 30(FAB)                             ; 0409
                                                03  11 00371        BRB     36$                                     ;
                                      1E        AB  94 00373 35$:    CLRB    30(FAB)                                 ; 0412
                                55      FF65    CD  9E 00376 36$:    MOVAB   HDR2+5, R5                              ; 0417
                                54              05  D0 0037B        MOVL    #5, R4                                  ;
                                53              0A  D0 0037E        MOVL    #10, R3                                 ;
                                   00000000G    EF  16 00381        JSB     SCAN$NUMERIC                            ;
                                50      01FF    C0  9E 00387        MOVAB   511(R0), R0                             ;
                                50 00000200     8F  C6 0038C        DIVL2   #512, R0                                ;
                                   00000000G    EF  16 00393        JSB     RMS$ALLOC_CACHE                         ;
                                66              50  D0 00399        MOVL    R0, (R6)                                ;
                                08              50  E8 0039C        BLBS    R0, 37$                                 ;
                                50 000184D4     8F  D0 0039F        MOVL    #99540, R0                              ; 0420
                                                04  00 3A6          RET                                             ;
                                         44     A9  D4 003A7 37$:    CLRL    68(CBLOCK)                              ; 0422
                                         42     A9  B4 003AA        CLRW    66(CBLOCK)                              ; 0423
                                24      A9             01  D0 003AD        MOVL    #1, 36(CBLOCK)                  ; 0424
                                28      A9             01  D0 003B1        MOVL    #1, 40(CBLOCK)                  ;
                                58      A9             01  D0 003B5        MOVL    #1, 88(CBLOCK)                  ; 0425
                         00000000G      EF             00  FB 003B9 38$:    CALLS   #0, GET_BLOCK_LENGTH            ; 0427
                                53              52  D0 003C0        MOVL    R2, R3                                  ;
                                68              50  E9 003C3        BLBC    R0, 41$                                 ;
                                01      58      A9  D1 003C6        CMPL    88(CBLOCK), #1                          ; 0433
                                                15  12 003CA        BNEQ    39$                                     ;
                                50      34      A9  D0 003CC        MOVL    52(CBLOCK), TMP                         ; 0440
                             34 A9      2C      A9  D0 003D0        MOVL    44(CBLOCK), 52(CBLOCK)                  ; 0441
                             2C A9              50  D0 003D5        MOVL    TMP, 44(CBLOCK)                         ; 0442
                             24 A9      0A      A9  3C 003D9        MOVZWL  10(CBLOCK), 36(CBLOCK)                  ; 0443
                                        24      A9  D6 003DE        INCL    36(CBLOCK)                              ;
                                50      0A      A9  3C 003E1 39$:    MOVZWL  10(CBLOCK), R0                          ; 0446
                                51      01      A0  9E 003E5        MOVAB   1(R0), R1                               ;
                                51      58      A9  D1 003E9        CMPL    88(CBLOCK), R1                          ;
                                                15  12 003ED        BNEQ    40$                                     ;
                                51      34      A9  D0 003EF        MOVL    52(CBLOCK), TMP                         ; 0453
                             34 A9      30      A9  D0 003F3        MOVL    48(CBLOCK), 52(CBLOCK)                  ; 0454
                             30 A9              51  D0 003F8        MOVL    TMP, 48(CBLOCK)                         ; 0455
                      24 A9                     50  78 003FC        ASHL    #1, R0, 36(CBLOCK)                      ; 0456
                                        24      A9  D6 00401        INCL    36(CBLOCK)                              ;
                             58 A9              50  C0 00404 40$:    ADDL2   R0, 88(CBLOCK)                          ; 0459
                         51                 50      44  A9  C1 00408        ADDL3   68(CBLOCK), R0, R1              ; 0460
                                                    51  D7 0040D        DECL    R1                                      ;
                                                    50  D7 0040F        DECL    R0                                      ;
                                                51  50  CA 00411        BICL2   R0, R1                                  ;
                                                52  01FF C3 9E 00414        MOVAB   511(R3), R2                     ; 0461
                         52                     52  F7  8F 78 00419        ASHL    #-9, R2, R2                     ;
                      44 A9                     51  52  C1 0041E        ADDL3   R2, R1, 68(CBLOCK)                      ;
                   50    53                     09  00  EF 00423        EXTZV   #0, #9, BLOCK_LENGTH, R0                ; 0462
                                         42     A9  50  B0 00428        MOVW    R0, 66(CBLOCK)                          ;
                                                    8B  11 0042C        BRB     38$                                     ; 0427
                                         44     A9  D6 0042E 41$:    INCL    68(CBLOCK)                              ; 0465
                                         38     A9  DD 00431        PUSHL   56(CBLOCK)                              ; 0466
                                                7E  25 7D 00434        MOVQ    #37, -(SP)                              ;
                                                7E  7C 00437        CLRQ    -(SP)                                   ;
                                                7E  01 CE 00439        MNEGL   #1, -(SP)                               ;
                                   00000000G    EF  06 FB 0043C        CALLS   #6, B00$QIO                             ;
```

```
                                66              50  D0 00443          MOVL    R0, (R6)                                ;
                                7E              66  E9 00446          BLBC    (R6), 47$                               ; 0468
                                50 00010001     8F  D0 00449          MOVL    #65537, R0                              ; 0470
                                                    04 00450          RET                                             ;
                     31524448   8F       B0    AD  D1 00451 42$:     CMPL    HDR1, #827475016                        ; 0475
                                                    03  13 00459          BEQL    43$                                 ;
                                                FC4A    31 0045B          BRW     4$                                  ;
                                54                  01  D0 0045E 43$:     MOVL    #1, R4                              ; 0479
                00AC    CE          B0      AD      0050    8F  29 00461          CMPC3   #80, HDR1, FIRST1                   ;
                                                            03  1A 00469          BGTRU   44$                             ;
                                54                          01  D9 0046C          SBWC    #1, R4                          ;
                                                            54  D5 0046F 44$:     TSTL    R4                              ;
                                                            71  12 00471          BNEQ    48$                             ;
                     34524448   8F       FF10            CD  D1 00473          CMPL    HDR4, #877806664                ; 0482
                                                            07  12 0047C          BNEQ    45$                             ;
                                                            58  DD 0047E          PUSHL   R8                              ; 0484
                                7E                          26  7D 00480          MOVQ    #38, -(SP)                      ;
                                                            05  11 00483          BRB     46$                             ;
                                                            58  DD 00485 45$:     PUSHL   R8                              ; 0488
                                7E                          25  7D 00487          MOVQ    #37, -(SP)                      ;
                                                            7E  7C 0048A 46$:     CLRQ    -(SP)                           ;
                                7E                          01  CE 0048C          MNEGL   #1, -(SP)                       ;
                     00000000G  EF                          06  FB 0048F          CALLS   #6, BOO$QIO                     ;
                                66                          50  D0 00496          MOVL    R0, (R6)                        ;
                                75                          66  E9 00499          BLBC    (R6), 50$                       ; 0489
                                                            58  DD 0049C          PUSHL   R8                              ; 0491
                                7E                          26  7D 0049E          MOVQ    #38, -(SP)                      ;
                                                            7E  7C 004A1          CLRQ    -(SP)                           ;
                                7E                          01  CE 004A3          MNEGL   #1, -(SP)                       ;
                     00000000G  EF                          06  FB 004A6          CALLS   #6, BOO$QIO                     ;
                                66                          50  D0 004AD          MOVL    R0, (R6)                        ;
                                74                          66  E9 004B0          BLBC    (R6), 52$                       ; 0492
                                                            58  DD 004B3          PUSHL   R8                              ; 0493
                                7E                          26  7D 004B5          MOVQ    #38, -(SP)                      ;
                                                            7E  7C 004B8          CLRQ    -(SP)                           ;
                                7E                          01  CE 004BA          MNEGL   #1, -(SP)                       ;
                     00000000G  EF                          06  FB 004BD          CALLS   #6, BOO$QIO                     ;
                                66                          50  D0 004C4          MOVL    R0, (R6)                        ;
                                76                          66  E9 004C7 47$:     BLBC    (R6), 53$                       ; 0494
                                                            58  DD 004CA          PUSHL   R8                              ; 0495
                                7E                          26  7D 004CC          MOVQ    #38, -(SP)                      ;
                                                            7E  7C 004CF          CLRQ    -(SP)                           ;
                                7E                          01  CE 004D1          MNEGL   #1, -(SP)                       ;
                     00000000G  EF                          06  FB 004D4          CALLS   #6, BOO$QIO                     ;
                                66                          50  D0 004DB          MOVL    R0, (R6)                        ;
                                7D                          66  E9 004DE          BLBC    (R6), 55$                       ; 0496
                                                        00A9    31 004E1          BRW     57$                             ;
                                            00AC        CE  95 004E4 48$:     TSTB    FIRST1                          ; 0501
                                                            09  12 004E8          BNEQ    49$                             ;
                00AC    CE          B0      AD      0050    8F  28 004EA          MOVC3   #80, HDR1, FIRST1               ;
                     34524448   8F       FF10            CD  D1 004F3 49$:     CMPL    HDR4, #877806664                ; 0503
                                                            16  12 004FC          BNEQ    51$                             ;
                                                            58  DD 004FE          PUSHL   R8                              ; 0505
                                7E                          25  7D 00500          MOVQ    #37, -(SP)                      ;
                                                            7E  7C 00503          CLRQ    -(SP)                           ;
                                                            01  DD 00505          PUSHL   #1                              ;
                     00000000G  EF                          06  FB 00507          CALLS   #6, BOO$QIO                     ;
```

```
                                 66     50  D0 0050E             MOVL    R0, (R6)                                ;
                                 4A     66  E9 00511 50$:       BLBC    (R6), 55$                               ; 0506
                                        58  DD 00514 51$:       PUSHL   R8                                      ; 0509
                                 7E     25  7D 00516             MOVQ    #37, -(SP)                              ;
                                        7E  7C 00519             CLRQ    -(SP)                                   ;
                                        01  DD 0051B             PUSHL   #1                                      ;
                          00000000G     EF  06  FB 0051D         CALLS   #6, B00$QIO                             ;
                                 66     50  D0 00524             MOVL    R0, (R6)                                ;
                                 76     66  E9 00527 52$:       BLBC    (R6), 61$                               ; 0511
                                        58  DD 0052A             PUSHL   R8                                      ; 0513
                                 7E     25  7D 0052C             MOVQ    #37, -(SP)                              ;
                                        7E  7C 0052F             CLRQ    -(SP)                                   ;
                                        01  DD 00531             PUSHL   #1                                      ;
                          00000000G     EF  06  FB 00533         CALLS   #6, B00$QIO                             ;
                                 66     50  D0 0053A             MOVL    R0, (R6)                                ;
                                 5A     66  E8 0053D             BLBS    (R6), 59$                               ; 0515
                                        5E  11 00540 53$:       BRB     61$                                     ;
                          00000838      8F  66  D1 00542 54$:       CMPL    (R6), #2104                             ; 0524
                                        1B  12 00549             BNEQ    56$                                     ;
                                        58  DD 0054B             PUSHL   R8                                      ; 0526
                                 7E     25  7D 0054D             MOVQ    #37, -(SP)                              ;
                                        7E  7C 00550             CLRQ    -(SP)                                   ;
                                        01  DD 00552             PUSHL   #1                                      ;
                          00000000G     EF  06  FB 00554         CALLS   #6, B00$QIO                             ;
                                 66     50  D0 0055B             MOVL    R0, (R6)                                ;
                                 3F     66  E9 0055E 55$:       BLBC    (R6), 61$                               ; 0528
                                 5A     04  8A 00561             BICB2   #4, FLAG                                ; 0530
                                        37  11 00564             BRB     60$                                     ; 0254
                          00000870      8F  66  D1 00566 56$:       CMPL    (R6), #2160                             ; 0533
                                        31  12 0056D             BNEQ    61$                                     ;
                       27               5A  02  E1 0056F         BBC     #2, FLAG, 59$                           ; 0536
                                        58  DD 00573             PUSHL   R8                                      ; 0539
                                 7E     24  7D 00575             MOVQ    #36, -(SP)                              ;
                                        7E  7C 00578             CLRQ    -(SP)                                   ;
                                        7E  D4 0057A             CLRL    -(SP)                                   ;
                             00000000G  EF  06  FB 0057C         CALLS   #6, B00$QIO                             ;
                                 66     50  D0 00583             MOVL    R0, (R6)                                ;
                                 5A     04  8A 00586             BICB2   #4, FLAG                                ; 0540
                       08        5A     01  E1 00589             BBC     #1, FLAG, 58$                           ; 0542
                                 50 00018292   8F  D0 0058D 57$:       MOVL    #98962, R0                              ;
                                        04  00594             RET                                             ;
                                 5A     02  88 00595 58$:       BISB2   #2, FLAG                                ;
                                        03  11 00598             BRB     60$                                     ; 0536
                                 5A     04  88 0059A 59$:       BISB2   #4, FLAG                                ; 0546
                                     FB08  31 0059D 60$:       BRW     4$                                      ; 0254
                                 50 0001C0F4   8F  D0 005A0 61$:       MOVL    #114932, R0                             ; 0551
                                        04  005A7             RET                                             ; 0557

; Routine Size:  1448 bytes,    Routine Base:  CODE + 0000

;   0558  1
```

```
;    0559  1     %sbttl 'ansi$close'
;    0560  1
;    0561  1     global routine ansi$close : call_rms =
;    0562  1
;    0563  1     !++
;    0564  1     ! Functional description:
;    0565  1     !
;    0566  1     !       This routine is used to close a file. It positions the tape
;    0567  1     !       to the final end-of-file mark.
;    0568  1     !       This positioning is important since if the same file
;    0569  1     !       is opened again the tape will already be positioned to the EOF1
;    0570  1     !       record.
;    0571  1     !
;    0572  1     ! Formal parameters:    none
;    0573  1     !
;    0574  1     ! Implicit inputs:
;    0575  1     !
;    0576  1     !       FAB     Address of FAB
;    0577  1     !       cblock  Address of perfile data
;    0578  1     !
;    0579  1     ! Implicit outputs:
;    0580  1     !
;    0581  1     ! Routine value:
;    0582  1     !
;    0583  1     !       RMS completion codes
;    0584  1     !
;    0585  1     ! Side effects:
;    0586  1     !
;    0587  1     !       The tape may be moved
;    0588  1     !
;    0589  1     !--
;    0590  1
;    0591  2         begin
;    0592  2
;    0593  2         external register
;    0594  2             cblock = 9 : ref block [, byte],
;    0595  2             fab = 11 : ref $fab_decl;
;    0596  2
;    0597  2         external routine                                                    [06]
;    0598  2             boo$qio : addressing_mode (long_relative);                      [06]
;    0599  2
;    0600  2         bind
;    0601  2             status = fab [fab$l_stv];
;    0602  2
;    0603  2         kb_check ();
;    0604  2         status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0605  2
;    0606  2         if not .status then return rms$_ccf;
;    0607  2
;    0608  2         return rms$_normal;
;    0609  1         end;


                                05FC 00000          .ENTRY  ANSI$CLOSE, Save R2,R3,R4,R5,R6,R7,R8,R10    ; 0561
```

```
                                    00000000G  EF  16 00002          JSB     KB_CHECK                                ; 0603
                                           38  A9  DD 00008          PUSHL   56(CBLOCK)                              ; 0604
                                    7E         25  7D 0000B          MOVQ    #37, -(SP)                              ;
                                               7E  7C 0000E          CLRQ    -(SP)                                   ;
                                               01  DD 00010          PUSHL   #1                                      ;
                              00000000G  EF        06  FB 00012          CALLS   #6, BOO$QIO                             ;
                                    0C   AB        50  D0 00019          MOVL    R0, 12(FAB)                             ;
                                         08    0C  AB  E8 0001D          BLBS    12(FAB), 1$                             ; 0606
                                         50 0001C0DC  8F  D0 00021          MOVL    #114908, R0                             ;
                                                      04  00028          RET                                             ;
                                         50 00010001  8F  D0 00029 1$:      MOVL    #65537, R0                              ; 0608
                                                      04  00030          RET                                             ; 0609

; Routine Size:  49 bytes,    Routine Base:  CODE + 05A8

;   0610  1
```

```
;    0611  1      %sbttl 'ansi$cachevbn'
;    0612  1
;    0613  1      global routine ansi$CACHEvbn (vbn) : call_rms =
;    0614  1
;    0615  1      !++
;    0616  1      ! Functional description:
;    0617  1      !
;    0618  1      !       This routine is used to place the desired block in the cache
;    0619  1      !       If necessary the tape is re-positioned to the desired block
;    0620  1      !       and the cache updated to include the desired data
;    0621  1      !
;    0622  1      ! Formal parameters:
;    0623  1      !
;    0624  1      !       VBN     VBN of block desired in the cache
;    0625  1      !
;    0626  1      ! Implicit inputs:
;    0627  1      !
;    0628  1      !       cblock  Address of per-file area
;    0629  1      !
;    0630  1      ! Implicit outputs:
;    0631  1      !
;    0632  1      !       cblock  The cache may be updated
;    0633  1      !
;    0634  1      ! Routine value:
;    0635  1      !
;    0636  1      !       Success or failure RMS codes and I/O failure codes
;    0637  1      !
;    0638  1      ! Side effects:
;    0639  1      !
;    0640  1      !       The device and controller may be accessed to read data from the tape.
;    0641  1      !
;    0642  1      !--
;    0643  1
;    0644  2          begin
;    0645  2
;    0646  2          external register
;    0647  2              cblock = 9 : ref block [, byte];        !control region
;    0648  2
;    0649  2          external routine                                                    [06]
;    0650  2              boo$qio : addressing_mode (long_relative);                      [06]
;    0651  2
;    0652  2          local
;    0653  2              status;                                 !Status of boo$qio calls
;    0654  2
;    0655  2          if .vbn geq .cblock [ctl$l_eofvbn] then return ss$_endoffile;
;    0656  2
;    0657  3          if (.vbn lss .cblock [ctl$l_lvbn]) or (.vbn geq .cblock [ctl$l_hvbn])
;    0658  2          then
;    0659  2                                                  !This block is not in the cache
;    0660  3              begin
;    0661  3
;    0662  3              while .cblock [ctl$l_tapepos] gtr .vbn do
;    0663  4                  begin                           !Backup until the block found
;    0664  4                  kb_check ();
;    0665  4                  status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);
;    0666  4
;    0667  4                  if not .status then return .status;
```

```
;    0668  4
;    0669  4                cblock [ctl$l_tapepos] = .cblock [ctl$l_tapepos] - .cblock [ctl$w_cache_size];
;    0670  4                cblock [ctl$l_lvbn] = cblock [ctl$l_hvbn] = .cblock [ctl$l_tapepos];
;    0671  3                end;
;    0672  3
;    0673  3            until (.vbn geq .cblock [ctl$l_lvbn]) and (.vbn lss .cblock [ctl$l_hvbn]) do
;    0674  4                begin                                   !Read forward until desired data is in the cache
;    0675  4
;    0676  4                local
;    0677  4                    cache;                              !Address of cache block
;    0678  4
;    0679  4                kb_check ();
;    0680  4
;    0681  4                case (.cblock [ctl$l_hvbn] - .cblock [ctl$l_lvbn])/.cblock [ctl$w_cache_size] from 0 to 3 of
;    0682  4                    set
;    0683  4
;    0684  4                    [0] :
;    0685  4                        cache = .cblock [ctl$l_cache1];
;    0686  4
;    0687  4                    [1] :
;    0688  4                        cache = .cblock [ctl$l_cache2];
;    0689  4
;    0690  4                    [2] :
;    0691  4                        cache = .cblock [ctl$l_cache3];
;    0692  4
;    0693  4                    [3] :
;    0694  5                        begin                           !Rotate cache in use
;    0695  5                        cache = .cblock [ctl$l_cache1];
;    0696  5                        cblock [ctl$l_cache1] = .cblock [ctl$l_cache2];
;    0697  5                        cblock [ctl$l_cache2] = .cblock [ctl$l_cache3];
;    0698  5                        cblock [ctl$l_cache3] = .cache;
;    0699  5                        cblock [ctl$l_lvbn] = .cblock [ctl$l_lvbn] + .cblock [ctl$w_cache_size];
;    0700  4                        end;
;    0701  4                    tes;
;    0702  4
;    0703  4                ch$fill (%c'^', 12*.cblock [ctl$w_cache_size], .cache);
;    0704  4                status = boo$qio (.cache, .cblock [ctl$w_cache_size]*512, 0, io$_readlblk, 0,
;    0705  4                    .cblock [ctl$l_rpb]);
;    0706  4
;    0707  4                if not .status then return .status;
;    0708  4
;    0709  4                cblock [ctl$l_tapepos] = .cblock [ctl$l_tapepos] + .cblock [ctl$w_cache_size];
;    0710  4                cblock [ctl$l_hvbn] = min (.cblock [ctl$l_eofvbn] + 1,
;    0711  4                    .cblock [ctl$l_hvbn] + .cblock [ctl$w_cache_size]);
;    0712  3                end;
;    0713  3
;    0714  2            end;
;    0715  2
;    0716  2        ss$_normal
;    0717  1        end;


                                ODFC 00000          .ENTRY  ANSI$CACHEVBN, Save R2,R3,R4,R5,R6,R7,R8,-  ; 0613
                                                            R10,R11                                         ;
```

```
                                5E              04  C2 00002          SUBL2   #4, SP                                          ;
                                5B        04    AC  D0 00005          MOVL    VEN, R11                                        ; 0655
                            44  A9              5B  D1 00009          CMPL    R11, 68(CBLOCK)                                 ;
                                                06  19 0000D          BLSS    1$                                              ;
                                50      0870    8F  3C 0000F          MOVZWL  #2160, R0                                       ;
                                                04  00014            RET                                                     ;
                            28  A9              5B  D1 00015 1$:      CMPL    R11, 40(CBLOCK)                                 ; 0657
                                                06  19 00019          BLSS    2$                                              ;
                            24  A9              5B  D1 0001B          CMPL    R11, 36(CBLOCK)                                 ;
                                                4A  19 0001F          BLSS    6$                                              ;
                                57        58    A9  9E 00021 2$:      MOVAB   88(CBLOCK), R7                                  ; 0662
                                58        0A    A9  9E 00025          MOVAB   10(CBLOCK), R8                                  ; 0669
                                5A        24    A9  9E 00029          MOVAB   36(CBLOCK), R10                                 ; 0670
                                5B              67  D1 0002D 3$:      CMPL    (R7), R11                                       ; 0662
                                                30  15 00030          BLEQ    5$                                              ;
                                    00000000G   EF  16 00032          JSB     KB_CHECK                                        ; 0664
                                        38      A9  DD 00038          PUSHL   56(CBLOCK)                                      ; 0665
                                7E              26  7D 0003B          MOVQ    #38, -(SP)                                      ;
                                7E              7E  7C 0003E          CLRQ    -(SP)                                           ;
                                7E              01  CE 00040          MNEGL   #1, -(SP)                                       ;
                        00000000G   EF          06  FB 00043          CALLS   #6, B00$Q10                                     ;
                                6E              50  D0 0004A          MOVL    R0, STATUS                                      ;
                                03              6E  E8 0004D          BLBS    STATUS, 4$                                      ; 0667
                                                008B 31 00050         BRW     14$                                             ;
                                50              68  3C 00053 4$:      MOVZWL  (R8), R0                                        ; 0669
                                67              50  C2 00056          SUBL2   R0, (R7)                                        ;
                                6A              67  D0 00059          MOVL    (R7), (R10)                                     ; 0670
                            28  A9              6A  D0 0005C          MOVL    (R10), 40(CBLOCK)                               ;
                                                CB  11 00060          BRB     3$                                              ; 0662
                            28  A9              5B  D1 00062 5$:      CMPL    R11, 40(CBLOCK)                                 ; 0673
                                                08  19 00066          BLSS    7$                                              ;
                                6A              5B  D1 00068          CMPL    R11, (R10)                                      ;
                                                03  18 0006B 6$:      BGEQ    7$                                              ;
                                            0091    31 0006D          BRW     17$                                             ;
                                    00000000G   EF  16 00070 7$:      JSB     KB_CHECK                                        ; 0679
                        50          28      A9  C3 00076          SUBL3   40(CBLOCK), (R10), R0                               ; 0681
                                51              68  3C 0007B          MOVZWL  (R8), R1                                        ;
                                50              51  C6 0007E          DIVL2   R1, R0                                          ;
                            03  00              50  CF 00081          CASEL   R0, #0, #3                                      ;
                    001A        0014    000E        0008      00085 8$:      .WORD   9$-8$,-                                  ;
                                                                                  10$-8$,-                                      ;
                                                                                  11$-8$,-                                      ;
                                                                                  12$-8$                                        ;
                                56        2C    A9  D0 0008D 9$:      MOVL    44(CBLOCK), CACHE                               ; 0685
                                                20  11 00091          BRB     13$                                             ;
                                56        30    A9  D0 00093 10$:     MOVL    48(CBLOCK), CACHE                               ; 0688
                                                1A  11 00097          BRB     13$                                             ;
                                56        34    A9  D0 00099 11$:     MOVL    52(CBLOCK), CACHE                               ; 0691
                                                14  11 0009D          BRB     13$                                             ;
                                56        2C    A9  D0 0009F 12$:     MOVL    44(CBLOCK), CACHE                               ; 0695
                            2C  A9        30    A9  7D 000A3          MOVQ    48(CBLOCK), 44(CBLOCK)                          ; 0696
                            34  A9              56  D0 000A8          MOVL    CACHE, 52(CBLOCK)                               ; 0698
                                50              68  3C 000AC          MOVZWL  (R8), R0                                        ; 0699
                            28  A9              50  C0 000AF          ADDL2   R0, 40(CBLOCK)                                  ;
                                50              68  3C 000B3 13$:     MOVZWL  (R8), R0                                        ; 0703
                                50              0C  C4 000B6          MULL2   #12, R0                                         ;
                50      5E  8F                  6E  00  2C 000B9      MOVC5   #0, (SP), #94, R0, (CACHE)                      ;
```

```
                                                66     000BF
                                          38    A9  DD 000C0          PUSHL   56(CBLOCK)                                ; 0705
                                                21  7D 000C3          MOVQ    #33, -(SP)                               ; 0704
                                             7E     7E  D4 000C6      CLRL    -(SP)                                    ;
                                                50  68 3C 000C8      MOVZWL  (R8), R0                                 ;
                                             7E     50  09 78 000CB   ASHL    #9, R0, -(SP)                            ;
                                                56  DD 000CF          PUSHL   CACHE                                    ;
                                          00000000G EF  06 FB 000D1   CALLS   #6, B00$QIO                              ;
                                                6E  50 D0 000D8       MOVL    R0, STATUS                               ;
                                                04  6E E8 000DB       BLBS    STATUS, 15$                              ; 0707
                                                50  6E D0 000DE 14$:  MOVL    STATUS, R0                               ;
                                                    04 000E1          RET                                              ;
                                                5C  68 3C 000E2 15$:  MOVZWL  [illegible], R0                          ; 0709
                                                67  50 C0 000E5       ADDL2   R0, (R7)                                 ;
                                             51     44  A9  01 C1 000E8 ADDL3  #1, 68(CBLOCK), R1                      ; 0710
                                                50  68 3C 000ED      MOVZWL  (R8), R0                                 ; 0711
                                                50  6A C0 000F0      ADDL2   (R10), R0                                ;
                                                50  51 D1 000F3      CMPL    R1, R0                                   ;
                                                    03 15 000F6       BLEQ    16$                                      ;
                                                51  50 D0 000F8      MOVL    R0, R1                                   ;
                                                5A  51 D0 000FB 16$:  MOVL    R1, (R10)                                ; 0710
                                              FF61  31 000FE          BRW     5$                                       ; 0673
                                                50  01 D0 00101 17$:  MOVL    #1, R0                                   ; 0717
                                                    04 00104          RET                                              ;

; Routine Size:  261 bytes,    Routine Base:  CODE + 05D9

;   0718  1
```

```
;   0719  1     %sbttl 'ansi$read'
;   0720  1
;   0721  1     global routine ansi$read : call_rms =
;   0722  1
;   0723  1     !++
;   0724  1     ! Functional description:
;   0725  1     !
;   0726  1     !       This routine is used to read a block from an open file.
;   0727  1     !       It uses the block number in the RAB to select the next block
;   0728  1     !       to be read.  Data blocks read from the tape are cached so
;   0729  1     !       this may not require a tape access.  Most of the work to fill
;   0730  1     !       and update the cache is done by ANSI$cacheVBN.
;   0731  1     !
;   0732  1     ! Formal parameters:    none
;   0733  1     !
;   0734  1     ! Implicit inputs:
;   0735  1     !
;   0736  1     !       FAB     Address of FAB for file
;   0737  1     !       RAB     Address of RAB from caller
;   0738  1     !       cblock  Address of per-file area
;   0739  1     !
;   0740  1     ! Implicit outputs:
;   0741  1     !
;   0742  1     !       FAB     Unaffected
;   0743  1     !       RAB     The current position is updated
;   0744  1     !       cblock  The cache may be updates
;   0745  1     !
;   0746  1     ! Routine value:
;   0747  1     !
;   0748  1     !       RMS completion codes
;   0749  1     !
;   0750  1     ! Side effects:
;   0751  1     !
;   0752  1     !       The device and controller may be accessed to read data from the tape.
;   0753  1     !
;   0754  1     !--
;   0755  1
;   0756  2         begin
;   0757  2
;   0758  2         external register
;   0759  2             cblock = 9 : ref block [, byte],
;   0760  2             fab = 11 : ref $fab_decl,
;   0761  2             rab = 10 : ref $rab_decl;
;   0762  2
;   0763  2         external routine                                    !                   [06]
;   0764  2             boo$qio : addressing_mode (long_relative);      !                   [06]
;   0765  2
;   0766  2         local
;   0767  2             cache : ref vector [, byte],                    !Address of current cache block
;   0768  2             user_size,                                      !Size to be transfered
;   0769  2             user_address,                                   !Address to receive data
;   0770  2             vbn;                                            !Current desired VBN
;   0771  2
;   0772  2         bind
;   0773  2             status = rab [rab$l_stv];                       !Current status return value
;   0774  2
;   0775  2         vbn = (if .rab [rab$l_bkt] eql 0 then .cblock [ctl$l_nbp] else .rab [rab$l_bkt]);
```

```
;    0776  2          user_size = .rab [rab$w_usz];
;    0777  2          user_address = rab [rab$l_rbf] = .rab [rab$l_ubf];
;    0778  2          rab [rab$w_rsz] = 0;
;    0779  2          rab [rab$l_rfa0] = .vbn;
;    0780  2          rab [rab$w_rfa4] = 0;
;    0781  2          status = rms$_normal;
;    0782  2
;    0783  2          while .user_size gtr 0 do
;    0784  3              begin
;    0785  3
;    0786  3              local
;    0787  3                  cache_offset,                       !offset within cache
;    0788  3                  cache_section;                      !Index to current section of cache
;    0789  3
;    0790  3              status = ansi$cachevbn (.vbn);
;    0791  3
;    0792  3              if not .status then exitloop;
;    0793  3
;    0794  3              cache_offset = (.vbn - .cblock [ctl$l_lvbn]);
;    0795  3              cache_section = .cache_offset/.cblock [ctl$w_cache_size];
;    0796  3              cache_offset = (.cache_offset - .cache_section*.cblock [ctl$w_cache_size])*512;
;    0797  3              cache = .(cblock [ctl$l_cache1] + .cache_section*4);
;    0798  3              ch$move (min (512, .user_size), cache [.cache_offset], .user_address);
;    0799  3              user_address = .user_address + min (512, .user_size);
;    0800  3              rab [rab$w_rsz] = .rab [rab$w_rsz] + min (512, .user_size);
;    0801  3              user_size = .user_size - min (512, .user_size);
;    0802  3              vbn = .vbn + 1;
;    0803  2              end;
;    0804  2
;    0805  2          cblock [ctl$l_nbp] = .vbn;
;    0806  2
;    0807  2          if .status and %x'FFFF' neq 0
;    0808  2          then
;    0809  2              .status
;    0810  2          else
;    0811  2
;    0812  2              if .status eql ss$_endoffile then rms$_normal else rms$_rer
;    0813  2
;    0814  1          end;


                                    01FC 00000          .ENTRY  ANSI$READ, Save R2,R3,R4,R5,R6,R7,R8        ; 0721
                              5E        08 C2 00002     SUBL2   #8, SP                                      ;
                          0C  AA    9F  00005           PUSHAB  12(RAB)                                     ; 0773
                          38  AA    D5  00008           TSTL    56(RAB)                                     ; 0775
                                    07  12  0000B       BNEQ    1$                                          ;
                    04    AE        04  A9  D0  0000D   MOVL    4(CBLOCK), VBN                              ;
                                    05  11  00012       BRB     2$                                          ;
                    04    AE        38  AA  D0  00014 1$: MOVL    56(RAB), VBN                                ;
                          56        20  AA  3C  00019 2$: MOVZWL  32(RAB), USER_SIZE                         ; 0776
                          50        24  AA  D0  0001D   MOVL    36(RAB), R0                                 ; 0777
                    28    AA            50  D0  00021   MOVL    R0, 40(RAB)                                 ;
                    08    AE            50  D0  00025   MOVL    R0, USER_ADDRESS                            ;
                                    22  AA  B4  00029   CLRW    34(RAB)                                     ; 0778
```

```
                          10   AA        04   AE   D0 0002C          MOVL    VBN, 16(RAB)                                    ; 0779
                                         14   AA   B4 00031          CLRW    2C(RAB)                                         ; 0780
                          00   BE 00010001   8F   D0 00034          MOVL    #65537, @0(SP)                                  ; 0781
                                         56   D5 0003C 3$:       TSTL    USER_SIZE                                       ; 0783
                                         59   15 0003E           BLEQ    5$                                              ;
                                         04   AE   DD 00040          PUSHL   VBN                                             ; 0790
                        FEB3   CF              01   FB 00043          CALLS   #1, ANSI$CACHEVBN                               ;
                          00   BE              50   D0 00048          MOVL    R0, @0(SP)                                      ;
                                         49   00   BE   E9 0004C      BLBC    @0(SP), 5$                                      ; 0792
                  51      04   AE        28   A9   C3 00050          SUBL3   40(CBLOCK), VBN, CACHE_OFFSET                   ; 0794
                                         50   0A   A9   3C 00056      MOVZWL  10(CBLOCK), CACHE_SECTION                       ; 0795
                  50                     51        50   C7 0005A      DIVL3   CACHE_SECTION, CACHE_OFFSET, CACHE_SECTION      ;
                                         52   0A   A9   3C 0005E      MOVZWL  10(CBLOCK), R2                                  ; 0796
                                              52   50   C4 00062      MULL2   CACHE_SECTION, R2                               ;
                  52                     51        52   C3 00065      SUBL3   R2, CACHE_OFFSET, R2                            ;
                  51                     52        09   78 00069      ASHL    #9, R2, CACHE_OFFSET                            ;
                                         58   2C A940   D0 0006D      MOVL    44(CBLOCK)[CACHE_SECTION], CACHE                ; 0797
                                         57        56   D0 00072      MOVL    USER_SIZE, R7                                   ; 0798
                     00000200   8F              57   D1 00075      CMPL    R7, #512                                        ;
                                              05   15 0007C          BLEQ    4$                                              ;
                                         57   0200 8F   3C 0007E      MOVZWL  #512, R7                                        ;
                08   BE             6148        57   28 00083 4$:       MOVC3   R7, (CACHE_OFFSET)[CACHE], @USER_ADDRESS        ;
                          08   AE              57   C0 00089          ADDL2   R7, USER_ADDRESS                                ; 0799
                          22   AA              57   A0 0008D          ADDW2   R7, 34(RAB)                                     ; 0800
                                         56        57   C2 00091      SUBL2   R7, USER_SIZE                                   ; 0801
                                              04   AE   D6 00094          INCL    VBN                                             ; 0802
                                              A3   11 00097          BRB     3$                                              ; 0783
                          04   A9        04   AE   D0 00099 5$:       MOVL    VBN, 4(CBLOCK)                                  ; 0805
                                         05   00   BE   E9 0009E      BLBC    @0(SP), 6$                                      ; 0807
                                         50   00   BE   D0 000A2      MOVL    @0(SP), R0                                      ; 0809
                                                   04 000A6          RET                                                     ;
                     00000870   8F        00   BE   D1 000A7 6$:       CMPL    @0(SP), #2160                                   ; 0812
                                                   08 12 000AF          BNEQ    7$                                              ;
                                         50 00010001 8F   D0 000B1      MOVL    #65537, R0                                      ;
                                                   04 000B8          RET                                                     ;
                                         50 0001C0F4 8F   D0 000B9 7$:       MOVL    #114932, R0                                     ;
                                                   04 000C0          RET                                                     ; 0814

; Routine Size:  193 bytes,    Routine Base:  CODE + 06DE

;   0815  1
```

```
;    0816  1      %sbttl 'ansi$get'
;    0817  1
;    0818  1      global routine ansi$get : call_rms =
;    0819  1
;    0820  1      !++
;    0821  1      ! Functional description:
;    0822  1      !
;    0823  1      !       This routine transfers the next record from the tape into
;    0824  1      !       the callers buffer. Only sequential mode is allowed.  ansi$cachevbn
;    0825  1      !       may be called to read more data from the tape.  The routine does
;    0826  1      !       the necessary deblocking of records from ANS X3.27 format tapes.
;    0827  1      !
;    0828  1      ! Formal parameters:    none
;    0829  1      !
;    0830  1      ! Implicit inputs:
;    0831  1      !
;    0832  1      !       FAB     Address of related open file block
;    0833  1      !       RAB     Address of RAB
;    0834  1      !       cblock  Address of per-file area
;    0835  1      !
;    0836  1      ! Implicit outputs:
;    0837  1      !
;    0838  1      !       FAB     unaffected
;    0839  1      !       RAB     the current position is updated
;    0840  1      !       cblock  Various fields may be updated
;    0841  1      !
;    0842  1      ! Routine value:
;    0843  1      !
;    0844  1      !       RMS completion codes
;    0845  1      !
;    0846  1      ! Side effects:
;    0847  1      !
;    0848  1      !       The device and associated channels may be accessed to read the record.
;    0849  1      !
;    0850  1      !--
;    0851  1
;    0852  2          begin
;    0853  2
;    0854  2          external register
;    0855  2              cblock = 9 : ref block [, byte],        !Base of control block
;    0856  2              rab = 10 : REF $RAB_DECL,               !Base of current rab
;    0857  2              fab = 11 : ref $fab_DECL;               !Base of current fab
;    0858  2
;    0859  2          external routine                            !                          [06]
;    0860  2              boo$qio : addressing_mode (long_relative);      !                  [06]
;    0861  2
;    0862  2          local
;    0863  2              boff,                                   !Current byte offset in page
;    0864  2              vbn;                                    !Current desired VBN
;    0865  2
;    0866  2          bind
;    0867  2              status = rab [rab$l_stv];               !Status returned by calls
;    0868  2
;    0869  2          vbn = .cblock [ctl$l_vbn];                  !Fetch current virtual block
;    0870  2          boff = .cblock [ctl$w_byte];                !Fetch byte offset
;    0871  2
;    0872  2          if .rab [rab$b_rac] eql rab$c_seq           !If sequential access
```

```
;    0873  2          then                                    !Update position based on RFA
;    0874  3              begin
;    0875  3              vbn = .vbn + (.cblock [ctl$w_rec_size] + .boff)/(512*.cblock [ctl$w_cache_size]);
;    0876  3              boff = (.cblock [ctl$w_rec_size] + .boff) mod (512*.cblock [ctl$w_cache_size]);
;    0877  2              end;
;    0878  2
;    0879  2          do
;    0880  3              begin
;    0881  3
;    0882  3              local
;    0883  3                  cache : ref vector [, byte],        !Address of current cache block
;    0884  3                  cache_offset,                       !offset within cache
;    0885  3                  cache_section;                      !Index to current section of cache
;    0886  3
;    0887  3              if not (status = ansi$cachevbn (.vbn)) then exitloop;
;    0888  3
;    0889  3              cache_offset = .vbn - .cblock [ctl$l_lvbn];
;    0890  3              cache_section = .cache_offset/.cblock [ctl$w_cache_size];
;    0891  3              cache_offset = .cache_offset - .cache_section*.cblock [ctl$w_cache_size]*512;
;    0892  3              cache = .(cblock [ctl$l_cache1] + .cache_section*4);
;    0893  3              rab [rab$l_rbf] = .rab [rab$l_ubf];
;    0894  3
;    0895  3              case .fab [fab$b_rfm] from fab$c_fix to fab$c_vfc of
;    0896  3                  set
;    0897  3
;    0898  3                  [fab$c_fix] :
;    0899  4                      begin
;    0900  4
;    0901  5                      if (.boff + .rab [fab$w_mrs]) geq (.cblock [ctl$w_cache_size]*512)
;    0902  4                      then
;    0903  5                          begin
;    0904  5                          vbn = .vbn + .cblock [ctl$w_cache_size];
;    0905  5                          boff = 0;
;    0906  5                          status = 0
;    0907  5                          end
;    0908  4                      else
;    0909  5                          begin
;    0910  5                          cblock [ctl$w_rec_size] = .fab [fab$w_mrs];
;    0911  5
;    0912  5                          if .rab [rab$w_usz] lss .cblock [ctl$w_rec_size] - 4
;    0913  5                          then
;    0914  6                              begin
;    0915  6                              status = rms$_rtb;
;    0916  6                              rab [rab$w_rsz] = .rab [rab$w_usz];
;    0917  6                              end
;    0918  5                          else
;    0919  6                              begin
;    0920  6                              status = rms$_normal;
;    0921  6                              rab [rab$w_rsz] = .cblock [ctl$w_rec_size];
;    0922  5                              end;
;    0923  5
;    0924  5                          ch$move (.rab [rab$w_rsz], cache [.boff], .rab [rab$l_rbf]);
;    0925  5                          status = rms$_normal;
;    0926  4                          end;
;    0927  4
;    0928  3                  end;
;    0929  3
```

```
;    0930  3                [fab$c_var] :
;    0931  4                    begin
;    0932  4
;    0933  5                    if (.cache [.boff] lss %c'0') or (.cache [.boff] gtr %c'9')
;    0934  4                    then
;    0935  5                        begin                           !Not valid record go to next block
;    0936  5                        boff = 0;
;    0937  5                        vbn = .vbn + .cblock [ctl$w_cache_size];
;    0938  5                        status = 0;
;    0939  5                        end
;    0940  4                    else
;    0941  5                        begin                           !Valid record copy it
;    0942  5                        cblock [ctl$w_rec_size] = scan$numeric (10, 4, cache [.boff]);
;    0943  5
;    0944  5                        if .rab [rab$w_usz] lss .cblock [ctl$w_rec_size] - 4
;    0945  5                        then
;    0946  6                            begin
;    0947  6                            status = rms$_rtb;
;    0948  6                            rab [rab$w_rsz] = .rab [rab$w_usz];
;    0949  6                            end
;    0950  5                        else
;    0951  6                            begin
;    0952  6                            status = rms$_normal;
;    0953  6                            rab [rab$w_rsz] = .cblock [ctl$w_rec_size] - 4;
;    0954  5                            end;
;    0955  5
;    0956  5                        ch$move (.rab [rab$w_rsz], cache [.boff + 4], .rab [rab$l_rbf]);
;    0957  5                        status = rms$_normal;
;    0958  4                        end;
;    0959  4
;    0960  3                    end;
;    0961  3
;    0962  3                [fab$c_vfc] :
;    0963  4                    begin
;    0964  4
;    0965  5                    if (.cache [.boff] lss %c'0') or (.cache [.boff] gtr %c'9')
;    0966  4                    then
;    0967  5                        begin                           !Not valid record go to next block
;    0968  5                        boff = 0;
;    0969  5                        vbn = .vbn + .cblock [ctl$w_cache_size];
;    0970  5                        status = 0;
;    0971  5                        end
;    0972  4                    else
;    0973  5                        begin                           !Valid record copy it
;    0974  5                        cblock [ctl$w_rec_size] = scan$numeric (10, 4, cache [.boff]);
;    0975  5
;    0976  5                        if .rab [rab$w_usz] lss .cblock [ctl$w_rec_size] - .fab [fab$b_fsz] - 4
;    0977  5                        then
;    0978  6                            begin
;    0979  6                            status = rms$_rtb;
;    0980  6                            rab [rab$w_rsz] = .rab [rab$w_usz] - .rab [fab$b_fsz];
;    0981  6                            end
;    0982  5                        else
;    0983  6                            begin
;    0984  6                            status = rms$_normal;
;    0985  6                            rab [rab$w_rsz] = .cblock [ctl$w_rec_size] - .fab [fab$b_fsz] - 4;
;    0986  5                            end;
```

```
;    0987  5
;    0988  5                        if .rab [rab$l_rhb] neq 0
;    0989  5                        then
;    0990  5                            ch$move (.fab [fab$b_fsz], cache [.boff + 4],
;    0991  5                                .rab [rab$l_rhb]);
;    0992  5
;    0993  5                        ch$move (.rab [rab$w_rsz], cache [.boff + .fab [fab$b_fsz] + 4], .rab [rab$l_rbf]);
;    0994  5                        status = rms$_normal;
;    0995  4                        end;
;    0996  4
;    0997  3                    end;
;    0998  3
;    0999  3                [outrange] :
;    1000  3                    return rms$_org;
;    1001  3                tes;
;    1002  3
;    1003  3            end
;    1004  2        until .status;
;    1005  2
;    1006  2        cblock [ctl$l_vbn] = .vbn;
;    1007  2        cblock [ctl$w_byte] = .boff;
;    1008  2
;    1009  2        if .status and %x'FFFF' NEQ 0
;    1010  2        then
;    1011  2            rms$_normal
;    1012  2        else
;    1013  2
;    1014  2            if .status eql ss$_endoffile then rms$_eof else rms$_rer
;    1015  2
;    1016  1        end;
```

```
                                    01FC 00000          .ENTRY  ANSI$GET, Save R2,R3,R4,R5,R6,R7,R8          ; 0818
                                5E      04   C2 00002          SUBL2   #4, SP                               ;
                                58  0C  AA   9E 00005          MOVAB   12(RAB), R8                          ; 0867
                                    18  A9   DD 00009          PUSHL   24(CBLOCK)                           ; 0869
                                56  1C  A9   3C 0000C          MOVZWL  28(CBLOCK), BOFF                     ; 0870
                                    1E  AA   95 00010          TSTB    30(RAB)                              ; 0872
                                        20   12 00013          BNEQ    1$                                   ;
                                52  1E  A9   3C 00015          MOVZWL  30(CBLOCK), R2                       ; 0875
                                52      56   C0 00019          ADDL2   BOFF, R2                             ;
                                50  0A  A9   3C 0001C          MOVZWL  10(CBLOCK), R0                       ;
                        50      50      09   78 00020          ASHL    #9, R0, R0                           ;
                        51      52      50   C7 00024          DIVL3   R0, R2, R1                           ;
                                6E      51   C0 00028          ADDL2   R1, VBN                              ;
                7E      00      52      01   7A 0002B          EMUL    #1, R2, #0, -(SP)                    ; 0876
                56      56      8E      50   7B 00030          EDIV    R0, (SP)+, BOFF, BOFF                ;
                                6E      DD 00035 1$:           PUSHL   VBN                                  ; 0887
                        FDFE    CF      01   FB 00037          CALLS   #1, ANSI$CACHEVBN                    ;
                                68      50   D0 0003C          MOVL    R0, (R8)                             ;
                                03      50   E8 0003F          BLBS    R0, 2$                               ;
                                    015C 31   00042          BRW     21$                                  ;
                        53      6E  28  A9   C3 00045 2$:      SUBL3   40(CBLOCK), VBN, CACHE_OFFSET        ; 0889
                                51  0A  A9   3C 0004A          MOVZWL  10(CBLOCK), R1                       ; 0890
```

```
                        50              53              51  C7 0004E            DIVL3   R1, CACHE_OFFSET, CACHE_SECTION                 ;
                        52              50              51  C5 00052            MULL3   R1, CACHE_SECTION, R2                     ; 0891
                        52              52              09  78 00056            ASHL    #9, R2, R2                                      ;
                                        53              52  C2 0005A            SUBL2   R2, CACHE_OFFSET                                ;
                                        57          2C A940 D0 0005D            MOVL    44(CBLOCK)[CACHE_SECTION], CACHE                ; 0892
                                28      AA          24  AA D0 00062            MOVL    36(RAB), 40(RAB)                                ; 0893
                            02          01          1F  AB 8F 00067            CASEB   31(FAB), #1, #2                                 ; 0895
                        00AF            005D            000E     0006C 3$:      .WORD   4$-3$,-                                         ;
                                                                                        8$-3$,-                                         ;
                                                                                        12$-3$                                          ;
                                        50 0001860C     8F  D0 00072            MOVL    #99852, R0                                      ; 1000
                                                            04 00079            RET                                                     ;
                                        52          36  AA 3C 0007A 4$:         MOVZWL  54(RAB), R2                                     ; 0901
                                        52              56  C0 0007E            ADDL2   BOFF, R2                                        ;
                            50          51              09  78 00081            ASHL    #9, R1, R0                                      ;
                                        50              52  D1 00085            CMPL    R2, R0                                          ;
                                                        08  19 00088            BLSS    5$                                              ;
                                        6E              51  C0 0008A            ADDL2   R1, VBN                                         ; 0904
                                                        56  D4 0008D            CLRL    BOFF                                            ; 0905
                                                    009A    31 0008F            BRW     14$                                             ; 0906
                                28  A9          36  AB B0 00092 5$:         MOVW    54(FAB), 30(CBLOCK)                             ; 0910
                                        50          1E  A9 3C 00097            MOVZWL  30(CBLOCK), R0                                  ; 0912
                                        50              04  C2 0009B            SUBL2   #4, R0                                          ;
                50      20  AA              10              00  ED 0009E            CMPZV   #0, #16, 32(RAB), R0                            ;
                                                            0E  18 000A4            BGEQ    6$                                              ;
                                        68 000181A8     8F  D0 000A6            MOVL    #98728, (R8)                                    ; 0915
                                22      AA          20  AA B0 000AD            MOVW    32(RAB), 34(RAB)                                ; 0916
                                                            0C  11 000B2            BRB     7$                                              ; 0912
                                        68 00010001     8F  D0 000B4 6$:         MOVL    #65537, (R8)                                    ; 0920
                                22      AA          1E  A9 B0 000BB            MOVW    30(CBLOCK), 34(RAB)                             ; 0921
                        28  BA              6647        22  AA 28 000C0 7$:         MOVC3   34(RAB), (BOFF)[CACHE], @40(RAB)        ; 0924
                                                            50  11 000C7            BRB     11$                                             ; 0925
                                        30              6647    91 000C9 8$:         CMPB    (BOFF)[CACHE], #48                      ; 0933
                                                            58  1F 000CD            BLSSU   13$                                             ;
                                        39              6647    91 000CF            CMPB    (BOFF)[CACHE], #57                              ;
                                                            52  1A 000D3            BGTRU   13$                                             ;
                            55          57              56  C1 000D5            ADDL3   BOFF, CACHE, R5                                 ; 0942
                                        54              04  D0 000D9            MOVL    #4, R4                                          ;
                                        53              0A  D0 000DC            MOVL    #10, R3                                         ;
                                    00000000G           EF  16 000DF            JSB     SCAN$NUMERIC                                    ;
                                1E  A9                  50  B0 000E5            MOVW    R0, 30(CBLOCK)                                  ;
                                        50          1E  A9 3C 000E9            MOVZWL  30(CBLOCK), R0                                  ; 0944
                                        50              04  C2 000ED            SUBL2   #4, R0                                          ;
                50      20  AA              10              00  ED 000F0            CMPZV   #0, #16, 32(RAB), R0                            ;
                                                            0E  18 000F6            BGEQ    9$                                              ;
                                        68 000181A8     8F  D0 000F8            MOVL    #98728, (R8)                                    ; 0947
                                22      AA          20  AA B0 000FF            MOVW    32(RAB), 34(RAB)                                ; 0948
                                                            0B  11 00104            BRB     10$                                             ; 0944
                                        68 00010001     8F  D0 00106 9$:         MOVL    #65537, (R8)                                    ; 0952
                                22      AA              50  B0 0010D            MOVW    R0, 34(RAB)                                     ; 0953
                        28  BA          04  A647        22  AA 28 00111 10$:         MOVC3   34(RAB), 4(BOFF)[CACHE], @40(RAB)       ; 0956
                                                            79  11 00119 11$:         BRB     19$                                             ; 0957
                                        30              6647    91 0011B 12$:         CMPB    (BOFF)[CACHE], #48                      ; 0965
                                                            06  1F 0011F            BLSSU   13$                                             ;
                                        39              6647    91 00121            CMPB    (BOFF)[CACHE], #57                              ;
                                                            09  18 00125            BLEQU   15$                                             ;
                                                            56  D4 00127 13$:         CLRL    BOFF                                    ; 0968
```

```
                               6E           51  C0 00129        ADDL2   R1, VBN                                    ; 0969
                                            68  D4 0012C  14$:  CLRL    (R8)                                       ; 0970
                                            6B  11 0012E        BRB     20$                                        ; 0965
                  55           57           56  C1 00130  15$:  ADDL3   BOFF, CACHE, R5                            ; 0974
                               54           04  D0 00134        MOVL    #4, R4                                     ;
                               53           0A  D0 00137        MOVL    #10, R3                                    ;
                                  00000000G EF  16 0013A        JSB     SCAN$NUMERIC                               ;
                          1E   A9           50  B0 00140        MOVW    R0, 30(CBLOCK)                             ;
                          04   AE        3F AB  9A 00144        MOVZBL  63(FAB), 4(SP)                             ; 0976
                               50        1E A9  3C 00149        MOVZWL  30(CBLOCK), R0                             ;
                               50        04 AE  C2 0014D        SUBL2   4(SP), R0                                  ;
                               50           04  C2 00151        SUBL2   #4, R0                                     ;
         50       20   AA      10           00  ED 00154        CMPZV   #0, #16, 32(RAB), R0                       ;
                                            13  18 0015A        BGEQ    16$                                        ;
                               68 000181A8  8F  D0 0015C        MOVL    #98728, (R8)                               ; 0979
                               50        3F AA  9A 00163        MOVZBL  63(RAB), R0                                ; 0980
                  22   AA      20   AA      50  A3 00167        SUBW3   R0, 32(RAB), 34(RAB)                       ;
                                            0B  11 0016D        BRB     17$                                        ; 0976
                               68 00010001  8F  D0 0016F  16$:  MOVL    #65537, (R8)                               ; 0984
                          22   AA           50  B0 00176        MOVW    R0, 34(RAB)                                ; 0985
                                         2C AA  D5 0017A  17$:  TSTL    44(RAB)                                    ; 0988
                                            08  13 0017D        BEQL    18$                                        ;
                  2C   BA      04 A647   04 AE  28 0017F        MOVC3   4(SP), 4(BOFF)[CACHE], @44(RAB)            ; 0991
                       50           56   04 AE  C1 00187  18$:  ADDL3   4(SP), BOFF, R0                            ; 0993
                  28   BA      04 A047   22 AA  28 0018C        MOVC3   34(RAB), 4(R0)[CACHE], @40(RAB)            ;
                               68 00010001  8F  D0 00194  19$:  MOVL    #65537, (R8)                               ; 0994
                               03           68  E8 0019B  20$:  BLBS    (R8), 21$                                  ; 1004
                                          FE94  31 0019E        BRW     1$                                         ;
                          18   A9           6E  D0 001A1  21$:  MOVL    VBN, 24(CBLOCK)                            ; 1006
                          1C   A9           56  B0 001A5        MOVW    BOFF, 28(CBLOCK)                           ; 1007
                               08           68  E9 001A9        BLBC    (R8), 22$                                  ; 1009
                               50 00010001  8F  D0 001AC        MOVL    #65537, R0                                 ;
                                                04 001B3        RET                                                ;
                     00000870  8F           68  D1 001B4  22$:  CMPL    (R8), #2160                                ; 1014
                                            08  12 001BB        BNEQ    23$                                        ;
                               50 0001827A  8F  D0 001BD        MOVL    #98938, R0                                 ;
                                                04 001C4        RET                                                ;
                               50 0001C0F4  8F  D0 001C5  23$:  MOVL    #114932, R0                                ;
                                                04 001CC        RET                                                ; 1016

; Routine Size:  461 bytes,    Routine Base:  CODE + 079F


;    1017  1
;    1018  1
;    1019  1    global routine get_block_length ( ;block_length) : get_block =  ! begin [06]
;    1020  1
;    1021  2        begin
;    1022  2
;    1023  2        linkage
;    1024  2            call_boo = call : global (r1 = 1);
;    1025  2
;    1026  2        external routine
;    1027  2            boo$qio : call_boo addressing_mode (long_relative);
;    1028  2
;    1029  2        global register
;    1030  2            r1 = 1;
```

```
;    1031  2
;    1032  2          external register
;    1033  2              cblock = 9 : ref block [, byte],              !Base of common area
;    1034  2              fab = 11 : ref $fab_decl;                     !Base of FAB
;    1035  2
;    1036  2          bind
;    1037  2              status = fab [fab$l_stv];                     !Return value from boo$qio
;    1038  2
;    1039  2          kb_check ();
;    1040  2          status = boo$qio (.cblock [ctl$l_cache3], .cblock [ctl$w_cache_size]*512, 0, io$_readlblk, 0, .cblock [ctl$l_rpb
;    1041  2          block_length = .r1;
;    1042  2          .status
;    1043  1          end;                                              ! end [06]



                                          05F8 00000        .ENTRY  GET_BLOCK_LENGTH, Save R3,R4,R5,R6,R7,R8,-  ; 1019
                                                            R10                                                 ;
                              00000000G   EF   16 00002     JSB     KB_CHECK                                    ; 1039
                                    38    A9   DD 00008     PUSHL   56(CBLOCK)                                  ; 1040
                                 7E       21   7D 0000B     MOVQ    #33, -(SP)                                  ;
                                          7E   D4 0000E     CLRL    -(SP)                                       ;
                                 50       0A   A9   3C 00010  MOVZWL  10(CBLOCK), R0                            ;
                              7E          50   09   78 00014  ASHL    #9, R0, -(SP)                               ;
                                    34    A9   DD 00018     PUSHL   52(CBLOCK)                                  ;
                           00000000G      EF   06   FB 0001B  CALLS   #6, BOO$QIO                                ;
                                 0C       AB   50   D0 00022  MOVL    R0, 12(FAB)                                 ;
                                 52       51   D0 00026     MOVL    R1, BLOCK_LENGTH                            ; 1041
                                 50       0C   AB   D0 00029  MOVL    12(FAB), R0                                 ; 1043
                                          04 0002D           RET                                                 ;

; Routine Size:  46 bytes,    Routine Base:  CODE + 096C


;    1044  1
;    1045  1      end
;    1046  0      eludom



;                                   PSECT SUMMARY
;
;        Name                    Bytes                           Attributes
;
;  DATA                            24  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          2458  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                               Library Statistics
;
```

```
;                                        -------- Symbols ---------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265              784         1         0          85        00:00.2
;  DRB1:[DS.WORK]DS.L32;159                653        23         3          42        00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7         18017        65         0         975        00:04.9




;                                    COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE ANSI

; Size:          2458 code + 24 data bytes
; Run Time:        00:47.2
; Elapsed Time:    03:32.9
; Lines/CPU Min:    1330
; Lexemes/CPU-Min: 16626
; Memory Used:  403 pages
; Compilation Complete
```

Table of contents

(1) 73 DECLARATIONS
(2) 102 APT SUPPORT ROUTINES

```
0000     1          .TITLE  APT     *** APT interface to APT control
0000     2          .IDENT  /07-10/
0000     3          .LIST   MEB
0000     4          .NLIST  CND
0000     5 ;
0000     6 ; COPYRIGHT (C) 1977, 1983
0000     7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     8 ;
0000     9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    15 ; REMAIN IN DEC.
0000    16 ;
0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    19 ; CORPORATION.
0000    20 ;
0000    21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    23
0000    24 ;++
0000    25 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    26 ;
0000    27 ; ABSTRACT:
0000    28 ;
0000    29 ; ENVIRONMENT:
0000    30 ;
0000    31 ; AUTHOR:          N. HOWGATE      03-DEC-77       VERSION 01.
0000    32 ; MODIFIED BY:
0000    33 ;                  N. HOWGATE      22-JUN-78       VERSION 02(ESSAA-4.03).
0000    34 ;       01         CORRECT DUMP-MODE PROBLEM
0000    35 ;                  N. HOWGATE      07-DEC-78       VERSION 03 (ESSAA-5.01)
0000    36 ;       02         EMULATE CHARACTER STRING INSTRUCTION
0000    37 ;                  R. RIGGS        12-JUL-79
0000    38 ;       03         ADD CODE TO EXECUTE APT PTEXT DECODER
0000    39 ;                  Roger Riggs, 28-May-1980, Version 5.4
0000    40 ;       04         Added code to check that on APT PTEXT command
0000    41 ;                  that there is at least one command to process.
0000    42 ;
0000    43 ;       05         - Jack Stansbury, 23-Oct-1981, Version 6.-
0000    44 ;                  Changed SEP Psect to CODE.
0000    45 ;                  Fixed truncation errors. Change to lower case on message.
0000    46 ;                  Also added .LIBRARY statements for $DS and $DIAG.
0000    47 ;
0000    48 ;       06         - Jack Stansbury, 12-Oct-1981, Version 6.-
0000    49 ;                  Changed the call to DS$PARSE to DSX$SUPERPARSE. See the CLI
0000    50 ;                  module for an explanation of SUPERPARSE. It returns with the
0000    51 ;                  low bit clear in R0 if anything went wrong.
0000    52 ;
0000    53 ;       07         - Jack Stansbury, 12-Oct-1981, Version 6.-
0000    54 ;                  Took out references to HALTD and HALTI. Made them just HALT.
0000    55 ;
0000    56 ;       08         - Jack Stansbury, 3-Feb-1982, Version 6.6
0000    57 ;                  Added a .LIBRARY statement for LIB. Note: This .LIBRARY
```

```
0000     58 ;                  statement must be the first one since the libraries are
0000     59 ;                  searched in LIFC (Last-In, First-Checked) order.
0000     60 ;
0000     61 ;          09      Bob Bergazzi    2-Feb-1983      Version 6.11
0000     62 ;                  Added code to COM_PTEXT that clears out DS$GL_CLIBASE.
0000     63 ;                  We should clear these flags out on every command,
0000     64 ;                  otherwise subsequent commands may be affected by previous
0000     65 ;                  commands. This coincides with a change to APT which allows
0000     66 ;                  any commands as PTEXT. The bug prompting this was
0000     67 ;                  SET EVENT 1, CLEAR EVENT 1, which caused a hung load.
0000     68 ;
0000     69 ;          10      Bob Bergazzi    11-Apr-1983     Version 6.11
0000     70 ;                  Fixed the previous edit.
0000     71 ;--
```

```
                        0000     73          .SBTTL  DECLARATIONS
                        0000     74 ;
                        0000     75 ; INCLUDE FILES:
                        0000     76 ;
                        0000     77          .LIBRARY        /SYS$LIBRARY:LIB/                       ;       [08]
                        0000     78          .LIBRARY        /$DS/                                   ;       [05]
                        0000     79          .LIBRARY        /$DIAG/                                 ;       [05]
                        0000     80
                        0000     81 ;
                        0000     82 ; MACROS:
                        0000     83 ;
                        0000     84
                        0000     85 ;
                        0000     86 ; EQUATED SYMBOLS:
                        0000     87 ;
                        0000     88 $DS_DSDEF
                        0000     89 $DS_DSADEF                                          ; APT DEFINITIONS
                        0000     90 $DS_HDRDEF                                          ; HEADER DEFINITIONS
                        0000     91 CLIDEF                                              ; CLI DEFINITIONS
                        0000     92 DSFDEF
                        0000     93
                        0000     94 ;
                        0000     95 ; OWN STORAGE:
                        0000     96 ;
                        0000     97          .SAVE
                    00000000     98          .PSECT  Data, NoExe, Shr, NoWrt, Long                   ;       [05]
                        0000     99          MODNAM  APT
            54 50 41 00' 0000        $MODULE:        .ASCIC  \APT\
                        03  0000
64 65 76 72 65 73 65 52 20 3F 3F 00' 0004    100 T_PRGERR:       .ASCIC  "?? Reserved APT command received"      ;       [05]
64 6E 61 6D 6D 6F 63 20 54 50 41 20  0010
           64 65 76 69 65 63 65 72 20  001C
                                    20  0004
```

```
                     0025   102          .SBTTL  APT SUPPORT ROUTINES
                 00000000   103          .PSECT  Code, Exe, Shr, NoWrt, Long                    ;                [05]
                     0000   104 ;++
                     0000   105 ; FUNCTIONAL DESCRIPTION:
                     0000   106 ;
                     0000   107 ;
                     0000   108 ; CALLING SEQUENCE:
                     0000   109 ;
                     0000   110 ;       NONE
                     0000   111 ;
                     0000   112 ; INPUT PARAMETERS:
                     0000   113 ;
                     0000   114 ;       NONE
                     0000   115 ;
                     C000   116 ; IMPLICIT INPUTS:
                     0000   117 ;
                     0000   118 ;       NONE
                     0000   119 ;
                     0000   120 ; OUTPUT PARAMETERS:
                     0000   121 ;
                     0000   122 ;       NONE
                     0000   123 ;
                     0000   124 ; IMPLICIT OUTPUTS:
                     0000   125 ;
                     0000   126 ;       NONE
                     0000   127 ;
                     0000   128 ; COMPLETION CODES:
                     0000   129 ;
                     0000   130 ;       NONE
                     0000   131 ;
                     0000   132 ; SIDE EFFECTS:
                     0000   133 ;
                     0000   134 ;       NONE
                     0000   135 ;
                     0000   136 ;--
                     0000   137
                     0000   138 APT_COM::
                     0000   139          CASE    DSA$GL_APTCOM,LIMIT=#0,TYPE=L,<-
                     0000   140          COM_NOP,-                                  ; <000> NOP
                     0000   141          COM_RSRV,-                                 ; <001> RESERVED
                     0000   142          COM_PTEXT,-                                ; <002> PROCESS PTEXT
                     0000   143          COM_DUMP,-                                 ; <003> DUMP
                     0000   144          COM_ZERO,-                                 ; <004> ZERO CORE
                     0000   145          COM_START,-                                ; <005> START DIAGNOSTIC
                     0000   146          COM_ABORT,-                                ; <006> ABORT DIAGNOSTIC
                     0000   147          COM_RSRV,-                                 ; <007> RESERVED
                     0000   148          COM_RSRV,-                                 ; <010> RESERVED
                     0000   149          COM_CONT,-                                 ; <011> CONTINUE
                     0000   150          >
09'  00    0000FE04'EF   CF  0000          CASEL   DSA$GL_APTCOM,#0,S^#<<30001$-30000$>/2>-1
                             0008      30000$:
                  0036'      0008          .SIGNED_WORD    COM_NOP-30000$
                  0014'      000A          .SIGNED_WORD    COM_RSRV-30000$
                  00DD'      000C          .SIGNED_WORD    COM_PTEXT-30000$
                  0043'      000E          .SIGNED_WORD    COM_DUMP-30000$
                  007B'      0010          .SIGNED_WORD    COM_ZERO-30000$
                  00AA'      0012          .SIGNED_WORD    COM_START-30000$
```

```
                        00BD' 0014                .SIGNED_WORD    COM_ABORT-30000$
                        0014' 0016                .SIGNED_WORD    COM_RSRV-30000$
                        0014' 0018                .SIGNED_WORD    COM_RSRV-30000$
                        00D0' 001A                .SIGNED_WORD    COM_CONT-30000$
                              001C          30001$:
                              001C    151
                              001C    152 COM_RSRV:
                0000FE04'EF    D4  001C    153         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
                0000FE40'EF    D4  0022    154         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
                                   0028    155         ERRSUP_S        MSGADR=T_PRGERR ; RESERVED COMMAND CODE
                00000000'EF    DF  0028                        PUSHAL  $MODULE
                00000004'EF    DF  002E                        PUSHAL  T_PRGERR
                          01   DD  0034                        PUSHL   #$ER
             00000000'EF   03   FB  0036                        CALLS   #3, DS_ERRSUP
                              05  C03D    156         RSB                                     ; RETURN
                              003E    157
                              003E    158 COM_NOP:
                0000FE04'EF    D4  003E    159         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
                0000FE40'EF    D4  0044    160         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
                              05   004A    161         RSB                                     ; RETURN
                              004B    162
                              004B    163 COM_DUMP:
                0000FE04'EF    D4  004B    164         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
                0000FE40'EF    D4  0051    165         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
    0000FE00'EF 88000000 8F    CA  0057    166         BICL    #DSA$M_APT!DSA$M_NORPT,DSA$GL_FLAGS     ; CLEAR APT AND NO REPORT
    0000FE00'EF 00001000 8F    C8  0062    167         BISL    #DSA$M_OPER,DSA$GL_FLAGS        ; INDICATE OPERATOR PRESENT
                          06   C8  006D    168         BISL    #DS$M_LODFLG!DS$M_STRFLG,-
                00000000'EF       006F    169                 DS$GL_FLAGS                     ; INDICATE PROGRAM LOADED AND STARTE
          04 A2 00000000'EF   9E  0074    170         MOVAB   VRRESTART,CLI$L_COMMAND(R2)     ; LOAD DUMMY RESTART COMMAND
                00000000'EF   16  007C    171         JSB     VRRESTART                       ; DO A RESTART                  [05]
                              05  0082    172         RSB                                     ; RETURN
```

```
                                0083   174 COM_ZERO:
00000000'8F   00    DA  0083   175         MTPR    #0,#PR$_MAPEN                   ; TURN OFF MEMORY MANAGEMENT IF ENAB
     0000FE04'EF    D4  008A   176         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
     0000FE40'EF    D4  0090   177         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
0000001C'EF   00    D2  0096   178         MCOML   #0,DS$GL_CLIBASE+CLI$L_DATA     ; SET UP AN "ALL" INDICATOR
     00000000'EF    16  009D   179         JSB     VRCLRBRK                        ;  AND GO CLEAR BREAKPOINTS     [05]
               50   D4  00A3   180         CLRL    R0                              ; CLEAR INDEX
                    00A5   181
          0000'C0   7C  00A5   182 10$:    CLRQ    DS$A_PRGBGN(R0)                 ; CLEAR
F4 50  00000000'8F  F2  00A9   183         AOBLSS  #<DS$K_PRGSIZ/8>,R0,10$         ; ALL LOCATIONS
               05   00B1   184         RSB                                     ; RETURN
                    00B2   185
                    00B2   186 COM_START:
     0000FE04'EF    D4  00B2   187         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
     0000FE40'EF    D4  00B8   188         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
     00000000'EF    16  00BE   189         JSB     VRSTART                         ; START                         [05]
               05   00C4   190         RSB                                     ; RETURN
                    00C5   191
                    00C5   192 COM_ABORT:
     0000FE04'EF    D4  00C5   193         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
     0000FE40'EF    D4  00CB   194         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
     00000000'EF    16  00D1   195         JSB     VRABORT                         ; ABORT                         [05]
               05   00D7   196         RSB                                     ; RETURN
                    00D8   197
                    00D8   198 COM_CONT:
     0000FE04'EF    D4  00D8   199         CLRL    DSA$GL_APTCOM                   ; CLEAR COMMAND
     0000FE40'EF    D4  00DE   200         CLRL    DSA$GL_MSGTYP                   ; CLEAR MESSAGE TYPE
               04   00E4   201         RET                                     ; RETURN TO CLI'S CALLER
```

```
                                00E5   203
                                00E5   204 COM_PTEXT:
                          30    BB  00E5   205          PUSHR   #^M<R4,R5>                      ; Save a couple
                55  0000FA00'EF    9E  00E7   206          MOVAB   DSA$AT_APTTXT,R5                ; Point to text area
                                00EE   207
                          04    11  00EE   208          BRB     15$                             ; Assume must be some text
                          65    95  00F0   209 10$:     TSTB    (R5)                            ; Any text left?
                          26    13  00F2   210          BEQL    25$                             ; NO, exit
                                00F4   211 15$:
                50  0000FE00'EF    9E  00F4   212          MOVAB   DSA$AT_APTTXT+1024,R0           ; Point past two pages
                        54    65    9E  00FB   213          MOVAB   (R5),R4                         ; Point to current position
                    64  0A0D  8F    B1  00FE   214 20$:     CMPW    #^X0A0D,(R4)                    ; CRLF here?
                          18    13  0103   215          BEQL    30$                             ; Branch if it is
                       F5 54    50    F2  0105   216          AOBLSS  R0,R4,20$                       ; increment and try next
                                0109   217          ERRSUP_S
                    00000000'EF     DF  0109                   PUSHAL  $MODULE
                                00     DD  010F                   PUSHL   #0
                                02     DD  0111                   PUSHL   #$ER
            00000000'EF         03     FB  0113                   CALLS   #3, DS_ERRSUP
                          0090  31    BRW  011A   218 25$:     BRW     100$                            ; Exit
                                011D   219
                        54    55    C2   011D   220 30$:     SUBL2   R5,R4                           ; Length of text just scanned
                        54    04    C2   0120   221          SUBL    #4,R4                           ; Must be at least 4 characters
                          13    18  0123   222          BGEQ    40$                             ; Branch if enough
                                0125   223          ERRSUP_S
                    00000000'EF     DF  0125                   PUSHAL  $MODULE
                                00     DD  012B                   PUSHL   #0
                                03     DD  012D                   PUSHL   #$ER
            00000000'EF         03     FB  012F                   CALLS   #3, DS_ERRSUP
                          75    11  0136   224          BRB     100$                            ; Exit
                                0138   225
            85  203E5344  8F    D1  0138   226 40$:     CMPL    #^A'DS> ',(R5)+                 ; Prompt present?   'DS> '
                          13    13  013F   227          BEQL    45$                             ; Branch if present
                                0141   228          ERRSUP_S                                    ; no prompt present
                    00000000'EF     DF  0141                   PUSHAL  $MODULE
                                00     DD  0147                   PUSHL   #0
                                04     DD  0149                   PUSHL   #$ER
            00000000'EF         03     FB  014B                   CALLS   #3, DS_ERRSUP
                          59    11  0152   229          BRB     100$                            ; Exit
                                0154   230
                          03    BB  0154   231 45$:     PUSHR   #^M<R0,R1>                      ; Better saved than sorry.          [09]
            50  00000444'EF    DE  0156   232          MOVAL   L^DS$GL_CLIBASE+CLI$K_SIZE,R0   ; Start at the end.              [09]
                          70    D4  015D   240          CLRL    -(R0)                           ; After this it is quad aligned     [10]
            51  00000088  8F    D0  015F   242          MOVL    #CLI$K_SIZE/8,R1                ; Number of QUAD words to clear     [09]
                                0166   243
                          70    7C  0166   244 50$:     CLRQ    -(R0)                           ; Loop til we reach beginning       [09]
                       FB 51    F5  0168   245          SOBGTR  R1,50$                          ;                                   [09]
                          03    BA  016B   246          POPR    #^M<R0,R1>                      ; Restore                           [09]
                                016D   247
                     34 A2      54  7D  016D   248          MOVQ    R4,CLI$Q_BUFQWD(R2)             ; Set length and address of input
                                0171   249
                    00000000'EF     DF  0171   250          PUSHAL  L^CLI_ACTION                    ; Push SUPERPARSE params            [06]
                    00000000'EF     DF  0177   251          PUSHAL  L^COMMAND_TREE                  ;                                   [06]
                          34 A2     7F  017D   252          PUSHAQ  CLI$Q_BUFQWD(R2)                ;                                   [06]
            00000000'9F         03     FB  0180   253          CALLS   #3, @#DSX$SUPERPARSE    ; Call the Supervisor Parser        [06]
                                0187   254
                       13 50    E8  0187   255          BLBS    R0,60$                          ; Branch if it worked               [06]
```

```
                                018A   256          ERRSUP_S                        ; OOPS, APT should not make mistakes
          00000000'EF    DF     018A                       PUSHAL  $MODULE
                  00     DD     0190                       PUSHL   #0
                  05     DD     0192                       PUSHL   #$ER
00000000'EF       03     FB     0194                       CALLS   #3, DS_ERRSUP
                  10     11     019B   257          BRB     100$                    ; Exit
                                019D   258
                                019D   259 ;                                                                          [06]
                                019D   260 ; No longer need to check NOTNUF flag.                                     [06]
                                019D   261 ;                                                                          [06]
                                019D   262
     50    04 A2     D0     019D   263 60$:    MOVL    CLI$L_COMMAND(R2),R0    ; Address of processing routine
                  02     13     01A1   264          BEQL    70$                     ; Branch if none
                  60     16     01A3   265          JSB     (R0)                    ; Call it
                                01A5   266
     55    02 A544   9E     01A5   267 70$:    MOVAB   2(R5)[R4],R5            ; Point past current line
                FF43     31     01AA   268          BRW     10$                     ; Try again
                                01AD   269
                  30     BA     01AD   270 100$:   POPR    #^M<R4,R5>              ; Restore
          0000FE04'EF    D4     01AF   271          CLRL    DSA$GL_APTCOM           ; Clear command
          0000FE40'EF    D4     01B5   272          CLRL    DSA$GL_MSGTYP           ; Clear error text
                  05     01BB   273          RSB                             ; Return
```

```
                        01BC    275
                        01BC    276 ;+
                        01BC    277 ; ROUTINE TO SYNCHRONIZE SUPERVISOR
                        01BC    278 ; WITH APT RECIEPT OF ERROR MESSAGES
                        01BC    279 ;-
                        01BC    280 APT_MSG::
0000FE00'EF   01   D3   01BC    281             BITL    #DSA$M_HALT, -                  ; IF HALT ON ERROR              [07]
                        01C3    282                     DSA$GL_FLAGS                    ;
                  0E    12    01C3    283             BNEQ    APT_RTN                 ; IN-LINE BPT HAS ALREADY
                        01C5    284                                                     ; BEEN SET
                        01C5    285 APT_DONE::
                        01C5    286 10$:
00000000'EF   16        01C5    287             JSB     KB_CHECK                        ; Check for ^C at terminal      [05]
0000FE40'EF   D5        01CB    288             TSTL    DSA$GL_MSGTYP                   ; WAIT FOR MESSAGE TO BE
                        01D1    289                                                     ; ACKNOWLEDGED
                  F2    12    01D1    290             BNEQ    10$                     ; LOOP
                        01D3    291 APT_RTN:
                        05    01D3    292             RSB                             ; RETURN
                        01D4    293 .END
```

```
$ER                             = 00000006    D
$MODULE                           00000000 R  D  02
APT_COM                           00000000 RG D  03
APT_DONE                          000001C5 RG D  03
APT_MSG                           000001BC RG D  03
APT_RTN                           000001D3 R  D  03
BIT...                          = 0000001A    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        00000444    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEST                        00000020    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
CLI_ACTION                        ********  X  03
COMMAND_TREE                      ********  X  03
COM_ABORT                         000000C5 R  D  03
COM_CONT                          000000D8 R  D  03
COM_DUMP                          0000004B R  D  03
COM_NOP                           0000003E R  D  03
COM_PTEXT                         000000E5 R  D  03
COM_RSRV                          0000001C R  D  03
COM_START                         000000B2 R  D  03
COM_ZERO                          000000E3 R  D  03
DS$A_PRGBGN                       ********  X  03
DS$GL_CLIBASE                     ********  X  03
DS$GL_FLAGS                       ********  X  03
DS$K_ERROR                      = 00000002    D
DS$K_NORMAL                     = 00000001    D
DS$K_PRGSIZ                       ********  X  03
DS$K_SEVERE                     = 00000004    D
DS$K_SUBSYS                     = 00000066    D
DS$K_WARNING                    = 00000000    D
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMRON                     = 00020000    D
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
```

```
DS$V_EXCEPT             =  00000013     D
DS$V_EXETST             =  00000012     D
DS$V_HLTFLG             =  00000003     D
DS$V_LODFLG             =  00000001     D
DS$V_MEMMGT             =  0000000F     D
DS$V_OUTPUT             =  00000017     D
DS$V_RUBFLG             =  00000005     D
DS$V_SCRIPT             =  00000015     D
DS$V_SETIMR             =  00000019     D
DS$V_STRFLG             =  00000002     D
DS$V_SUBT               =  0000000E     D
DS$V_SYSFLG             =  0000000A     D
DS$V_TIMRON             =  00000011     D
DS$_ARITH               =  006600D0     D
DS$_ASBE                =  00660118     D
DS$_BADLINK             =  006600F0     D
DS$_BADTYPE             =  006600E8     D
DS$_BIIC                =  00660120     D
DS$_CHME                =  006600A8     D
DS$_CHMK                =  006600E0     D
DS$_DEVNAME             =  00660108     D
DS$_ERROR               =  00660002     D
DS$_FHWE                =  00660068     D
DS$_FRAGBUF             =  00660080     D
DS$_ICBUSY              =  006600C8     D
DS$_ICERR               =  006600C0     D
DS$_IHWE                =  00660060     D
DS$_ILLCHAR             =  00660018     D
DS$_ILLPAGCNT           =  00660078     D
DS$_ILLUNIT             =  00660100     D
DS$_INSFMEM             =  00660050     D
DS$_IPL2HI              =  006600B8     D
DS$_IVADDR              =  00660040     D
DS$_IVVECT              =  00660038     D
DS$_KRNLSTK             =  00660090     D
DS$_LOGIC               =  00660070     D
DS$_MCHK                =  00660088     D
DS$_MMOFF               =  00660058     D
DS$_NEEDUNIT            =  006600F8     D
DS$_NODF                =  00660128     D
DS$_NOPCS               =  00660110     D
DS$_NORMAL              =  00660001     D
DS$_NOSUPPORT           =  006600B1     D
DS$_NOTDON              =  00660030     D
DS$_NOTIMP              =  006600B0     D
DS$_NULLSTR             =  00660010     D
DS$_OVERFLOW            =  00660008     D
DS$_POWER               =  00660098     D
DS$_PROGERR             =  00660020     D
DS$_SEVERE              =  00660004     D
DS$_TRANSL              =  006600A0     D
DS$_TRUNCATE            =  00660028     D
DS$_UNEXPINT            =  006600D8     D
DS$_VASFULL             =  00660048     D
DS$_WARNING             =  00660000     D
DSA$AL_APTMAIL             0000FE00     D
DSA$AT_APTTXT              0000FA00     D
DSA$GL_APTCOM              0000FE04     D
DSA$GL_DEVLEN              0000FE58     D
DSA$GL_ERRNO               0000FE44     D
DSA$GL_EVENT               0000FE48     D
DSA$GL_FLAGS               0000FE00     D
DSA$GL_MSGTYP              0000FE40     D
DSA$GL_PASSES              0000FE08     D
DSA$GL_PASSNO              0000FE54     D
DSA$GL_SECTNO              0000FE10     D
DSA$GL_SID                 0000FE14     D
DSA$GL_SUBTNO              0000FE4C     D
DSA$GL_TESTNO              0000FE50     D
DSA$GL_UNITS               0000FE0C     D
DSA$GQ_MSGPTR              0000FE68     D
DSA$GT_DEVNAM              0000FE5C     D
DSA$M_APT               =  80000000     D
DSA$M_HALT              =  00000001     D
DSA$M_NORPT             =  08000000     D
DSA$M_OPER              =  00001000     D
DSX$SOPERPARSE             ********     X    03
DS_ERRSUP                  ********     X    03
KB_CHECK                   ********     X    03
L$A_CCP                    00000240     D
L$A_DEVP                   0000021C     D
L$A_DREG                   00000224     D
L$A_DTP                    00000218     D
L$A_ICP                    0000023C     D
L$A_LASTADR                0000C214     D
L$A_NAME                   00000208     D
L$A_REPP                   00000244     D
L$A_SECNAM                 00000250     D
L$A_STATAB                 00000248     D
L$A_TSTCNT                 00000254     D
L$L_ENVIRON                00000204     D
L$L_ERRTYP                 0000024C     D
L$L_HEADLENGTH             00000200     D
L$L_REV                    0000020C     D
L$L_UNIT                   00000220     D
L$L_UNUSED                 00000228     D
L$L_UPDATE                 00000210     D
PR$_MAPEN                  ********     X    03
SI7...                  =  00000001     D
T_PRGERR                   00000004 R   D    02
VRABORT                    ********     X    03
VRCLRBRK                   ********     X    03
VRRESTART                  ********     X    03
VRSTART                    ********     X    03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000025 (37.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	000001D4 (468.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000006	256 (5)	#-155 (2) #-217 (5) #-223 (5) #-228 (5) #-256 (5)
$MODULE	00000000-R	99 (1)	155 (2) 217 (5) 223 (5) 228 (5) 256 (5)
APT_COM	00000000-R	138 (2)	
APT_DONE	000001C5-R	285 (7)	
APT_MSG	000001BC-R	280 (7)	
APT_RTN	000001D3-R	291 (7)	#-283 (7)
BIT...	=0000001A	92 (1)	88 (1) 92 (1)
CLI$K_BUFSIZ	=00000100	91 (1)	91 (1)
CLI$K_SIZE	00000444	91 (1)	232 (5) 233 (5) 236 (5) 239 (5) #-242 (5)
CLI$L_ADDRESS	00000018	91 (1)	
CLI$L_COMMAND	00000004	91 (1)	#-170 (2) #-263 (5)
CLI$L_DATA	0000001C	91 (1)	#-178 (3)
CLI$L_FLAGS	00000000	91 (1)	
CLI$L_LAST	00000024	91 (1)	
CLI$L_NEXT	00000030	91 (1)	
CLI$L_PASS	0000002C	91 (1)	
CLI$L_SUBT	00000028	91 (1)	
CLI$L_TEST	00000020	91 (1)	
CLI$Q_BUFQWD	00000034	91 (1)	#-248 (5) 252 (5)
CLI$Q_FILE	00000008	91 (1)	
CLI$Q_SECTION	00000010	91 (1)	
CLI$Q_TIME	0000043C	91 (1)	
CLI$T_BUFFER	0000003C	91 (1)	
CLI$V_ADAPTER	=00000018	91 (1)	
CLI$V_ADR	=0000000B	91 (1)	
CLI$V_ASCII	=00000013	91 (1)	
CLI$V_BREAK	=0000000A	91 (1)	
CLI$V_BRIEF	=0000001B	91 (1)	
CLI$V_BYTE	=0000000D	91 (1)	
CLI$V_CLEAR	=00000002	91 (1)	
CLI$V_DEC	=00000010	91 (1)	
CLI$V_DEFAULT	=0000000C	91 (1)	
CLI$V_DEPOSIT	=00000019	91 (1)	
CLI$V_EVENT	=00000008	91 (1)	
CLI$V_EXAM	=00000005	91 (1)	
CLI$V_FLAGS	=00000009	91 (1)	
CLI$V_HEX	=00000012	91 (1)	
CLI$V_KERNEL	=00000017	91 (1)	
CLI$V_LOAD	=00000006	91 (1)	
CLI$V_LONG	=0000000F	91 (1)	
CLI$V_NOTNUF	=00000001	91 (1)	
CLI$V_OCT	=00000011	91 (1)	
CLI$V_PREG	=0000001A	91 (1)	
CLI$V_QA	=00000007	91 (1)	
CLI$V_QACKLOOPLOOPS	=0000001C	91 (1)	
CLI$V_QAERRORPRINTS	=0000001B	91 (1)	
CLI$V_QAMULTIPLEPASS	=0000001F	91 (1)	

```
CLI$V_QASUBTESTLOOPS            =0000001E      91      (1)
CLI$V_QATESTLOOPS               =0000001D      91      (1)
CLI$V_REG                       =00000014      91      (1)
CLI$V_REQUIRED                  =00000000      91      (1)
CLI$V_RUN                       =00000015      91      (1)
CLI$V_SET                       =00000003      91      (1)
CLI$V_SHOW                      =00000004      91      (1)
CLI$V_VALSEC                    =00000016      91      (1)
CLI$V_WORD                      =0000000E      91      (1)
CLI_ACTION                       00000000-XR                    250     (5)
COMMAND_TREE                     00000000-XR                    251     (5)
COM_ABORT                        000000C5-R     192     (3)     150     (2)
COM_CONT                         000000D8-R     198     (3)     150     (2)
COM_DUMP                         0000004B-R     163     (2)     150     (2)
COM_NOP                          00000C3E-R     158     (2)     150     (2)
COM_PTEXT                        000000E5-R     204     (5)     150     (2)
COM_RSRV                         0000001C-R     152     (2)     150     (2)
COM_START                        000000B2-R     186     (3)     150     (2)
COM_ZERO                         00000083-R     174     (3)     150     (2)
DS$A_PRGBGN                      00000000-XR                    #-182   (3)
DS$GL_CLIBASE                    00000000-XR                    #-178   (3)     232     (5)
DS$GL_FLAGS                      00000000-XR                    #-169   (2)
DS$K_ERROR                      =00000002      88      (1)
DS$K_NORMAL                     =00000001      88      (1)
DS$K_PRGSIZ                      00000000-XR                    #-183   (3)
DS$K_SEVERE                     =00000004      88      (1)
DS$K_SUBSYS                     =00000066      88      (1)     88      (1)
DS$K_WARNING                    =00000000      88      (1)
DS$M_ABRTFLG                    =00000040      92      (1)
DS$M_BADTIME                    =00100000      92      (1)
DS$M_BATCH                      =00400000      92      (1)
DS$M_BRKCLR                     =00001000      92      (1)
DS$M_BRKPT                      =00000800      92      (1)
DS$M_CHARFLG                    =00000100      92      (1)
DS$M_CMDFLG                     =00000080      92      (1)
DS$M_CTRLC                      =00000001      92      (1)
DS$M_CTRLO                      =00010000      92      (1)
DS$M_DEVFLG                     =00000200      92      (1)
DS$M_DISABLCC                   =01000000      92      (1)
DS$M_DONFLG                     =00002000      92      (1)
DS$M_ERRFLG                     =00000010      92      (1)
DS$M_EXCEPT                     =00080000      92      (1)
DS$M_EXETST                     =00040000      92      (1)
DS$M_HLTFLG                     =00000008      92      (1)
DS$M_LODFLG                     =00000002      92      (1)     #-168   (2)
DS$M_MEMMGT                     =00008000      92      (1)
DS$M_OUTPUT                     =00800000      92      (1)
DS$M_RUBFLG                     =00000020      92      (1)
DS$M_SCRIPT                     =00200000      92      (1)
DS$M_SETIMR                     =02000000      92      (1)
DS$M_STRFLG                     =00000004      92      (1)     #-168   (2)
DS$M_SUBT                       =00004000      92      (1)
DS$M_SYSFLG                     =00000400      92      (1)
DS$M_TIMRON                     =00020000      92      (1)
DS$V_ABRTFLG                    =00000006      92      (1)
DS$V_BADTIME                    =00000014      92      (1)
DS$V_BATCH                      =00000016      92      (1)
```

```
DS$V_BRKCLR                     =0000000C       92      (1)
DS$V_BRKPT                      =0000000B       92      (1)
DS$V_CHARFLG                    =00000008       92      (1)
DS$V_CMDFLG                     =00000007       92      (1)
DS$V_CTRLC                      =00000000       92      (1)
DS$V_CTRLO                      =00000010       92      (1)
DS$V_DEVFLG                     =00000009       92      (1)
DS$V_DISABLCC                   =00000018       92      (1)
DS$V_DONFLG                     =0000000D       92      (1)
DS$V_ERRFLG                     =00000004       92      (1)
DS$V_EXCEPT                     =00000013       92      (1)
DS$V_EXETST                     =00000012       92      (1)
DS$V_HLTFLG                     =00000003       92      (1)
DS$V_LODFLG                     =00000001       92      (1)
DS$V_MEMMGT                     =0000000F       92      (1)
DS$V_OUTPUT                     =00000017       92      (1)
DS$V_RUBFLG                     =00000005       92      (1)
DS$V_SCRIPT                     =00000015       92      (1)
DS$V_SETIMR                     =00000019       92      (1)
DS$V_STRFLG                     =00000002       92      (1)
DS$V_SUBT                       =0000000E       92      (1)
DS$V_SYSFLG                     =0000000A       92      (1)
DS$V_TIMRON                     =00000011       92      (1)
DS$_ARITH                       =006600D0       88      (1)
DS$_ASBE                        =00660118       88      (1)
DS$_BADLINK                     =006600F0       88      (1)
DS$_BADTYPE                     =006600E8       88      (1)
DS$_BIIC                        =00660120       88      (1)
DS$_CHME                        =006600A8       88      (1)
DS$_CHMK                        =006600E0       88      (1)
DS$_DEVNAME                     =00660108       88      (1)
DS$_ERROR                       =00660002       88      (1)
DS$_FHWE                        =00660068       88      (1)
DS$_FRAGBUF                     =00660080       88      (1)
DS$_ICBUSY                      =006600C8       88      (1)
DS$_ICERR                       =006600C0       88      (1)
DS$_IHWE                        =00660060       88      (1)
DS$_ILLCHAR                     =00660018       88      (1)
DS$_ILLPAGCNT                   =00660078       88      (1)
DS$_ILLUNIT                     =00660100       88      (1)
DS$_INSFMEM                     =00660050       88      (1)
DS$_IPL2HI                      =00660088       88      (1)
DS$_IVADDR                      =00660040       88      (1)
DS$_IVVECT                      =00660038       88      (1)
DS$_KRNLSTK                     =00660090       88      (1)
DS$_LOGIC                       =00660070       88      (1)
DS$_MCHK                        =00660088       88      (1)
DS$_MMOFF                       =00660058       88      (1)
DS$_NEEDUNIT                    =006600F8       88      (1)
DS$_NODE                        =00660128       88      (1)
DS$_NOPCS                       =00660110       88      (1)
DS$_NORMAL                      =00660001       88      (1)
DS$_NOSUPPORT                   =00660061       88      (1)
DS$_NOTDON                      =00660030       88      (1)
DS$_NOTIMP                      =006600B0       88      (1)     88      (1)
DS$_NULLSTR                     =00660010       88      (1)
DS$_OVERFLOW                    =00660008       88      (1)
```

```
DS$_POWER                       =00660098      88      (1)
DS$_PROGERR                     =00660020      88      (1)
DS$_SEVERE                      =00660004      88      (1)
DS$_TRANSL                      =006600A0      88      (1)
DS$_TRUNCATE                    =00660028      88      (1)
DS$_UNEXPINT                    =006600D8      88      (1)
DS$_VASFULL                     =00660048      88      (1)
DS$_WARNING                     =00660000      88      (1)      88      (1)
DSA$AT_APTTXT                    0000FA00                       206     (5)      212     (5)
DSA$GL_APTCOM                    0000FE04                     #-150     (2)    #-153     (2)    #-159     (2)    #-164     (2)
                                                              #-176     (3)    #-187     (3)    #-193     (3)    #-199     (3)
                                                              #-271     (5)
DSA$GL_FLAGS                     0000FE00                     #-166     (2)    #-167     (2)    #-282     (7)
DSA$GL_MSGTYP                    0000FE40                     #-154     (2)    #-160     (2)    #-165     (2)    #-177     (3)
                                                              #-188     (3)    #-194     (3)    #-200     (3)    #-272     (5)
                                                              #-288     (7)
DSA$M_APT                       =80000000                     #-166     (2)
DSA$M_HALT                      =00000001                     #-281     (7)
DSA$M_NORPT                     =08000000                     #-166     (2)
DSA$M_OPER                      =00001000                     #-167     (2)
DSX$SUPERPARSE                   00000000-XR                    253     (5)
DS_ERRSUP                        00000000-XR                    155     (2)      217     (5)      223     (5)      228     (5)
                                                                256     (5)
KB_CHECK                         00000000-XR                    287     (7)
PR$_MAPEN                        00000000-XR                  #-175     (3)
SIZ...                          =00000001      92      (1)      92     (1)
T_PRGERR                         00000C04-R    100      (1)     155     (2)
VRABORT                          00000000-XR                    195     (3)
VRCLRBRK                         00000000-XR                    179     (3)
VRRESTART                        00000000-XR                    170     (2)      171     (2)
VRSTART                          00000000-XR                    189     (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION		REFERENCES...									
$DEF	1	92	(1)										
$DEFINI	1	89	(1)	89	(1)	90	(1)						
$DS_DSADEF	5	89	(1)	89	(1)								
$DS_DSDEF	2	88	(1)	88	(1)								
$DS_HDRDEF	2	90	(1)	90	(1)								
$EQU	1	92	(1)	88	(1)								
$EQULS1	1	88	(1)	88	(1)								
$EQULST	1	88	(1)	88	(1)								
$GBLINI	2	88	(1)	88	(1)	92	(1)						
$PUSHADR	1	155	(2)	155	(2)	217	(5)	223	(5)	228	(5)	256	(5)
$VIELD	1			92	(1)								
$VIELD1	1	92	(1)	92	(1)								
CASE	1	139	(2)	139	(2)								
CLIDEF	3	91	(1)	91	(1)								
DSFDEF	3	92	(1)	92	(1)								
ERRSUP_S	1	155	(2)	155	(2)	217	(5)	223	(5)	228	(5)	256	(5)
MODNAM	1	99	(1)	99	(1)								

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.11	00:00:00.37
Command processing	140	00:00:00.71	00:00:01.63
Pass 1	698	00:00:06.90	00:00:11.00
Symbol table sort	0	00:00:00.35	00:00:00.35
Pass 2	100	00:00:01.29	00:00:02.07
Symbol table output	27	00:00:00.17	00:00:00.17
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	47	00:00:00.70	00:00:01.12
Assembler run totals	1053	00:00:10.29	00:00:16.76

The working set limit was 1000 pages.
68019 bytes (133 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 267 non-local and 15 local symbols.
293 source lines were read in Pass 1, producing 0 object records in Pass 2.
49 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	3
DRB1:[DS.WORK]DS.MLB;218	4
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	14

332 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) APT/UPDA=(APT.UPD,APT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DIAG/
```

Table of contents

(1) 92 ASSIGN I/O CHANNEL
(1) 356 TEST IF MAILBOX SPECIFIED

```
                        0000     .1          .TITLE  ASSIGN *** ASSIGN assign device
V54                     0000     .2          .IDENT  /05-02/
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        C000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 25-AUG-76
                        0000     27 ;
                        0000     28 ; SYSTEM SERVICE ASSIGN I/O CHANNEL
                        0000     29 ;
                        0000     30 ; MODIFICATION HISTORY:
V54                     0000     .1 ;
V54                     0000     .2 ;                    Dave Butenhof   13-may-1980, Version 5.4
V54                     0000     .3 ;       02           Modify VMS V2.0 source to run in supervisor environment
V54                     0000     .4 ;
V54                     0000     .5 ;                    N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
V54                     0000     .6 ;       01           DIAGNOSTIC SUPERVISOR INTEGRATION
                        0000     31 ;
                        0000     32 ;       V08          LMK0003         LEN KAWELL      20-JAN-1980
                        0000     33 ;                    ADD NON-SHAREABLE DEVICE IMPLICIT ALLOCATION PRIVILEGE CHECK.
                        0000     34 ;
                        0000     35 ;       V07          RIH0033         R. HUSTVEDT     16-OCT-1979
                        0000     36 ;                    CHANGE PCB$W_BYTCNT TO JIB$L_BYTCNT.
                        0000     37 ;
                        0000     38 ;       V06          LMK0002         LEN KAWELL      27-SEP-1979
                        0000     39 ;                    SET CCB$M_AMB IF A MAILBOX WAS ASSOCIATED WITH DEVICE.
                        0000     40 ;
                        0000     41 ;       V05          LMK0001         LEN KAWELL      27-JUL-1979
                        0000     42 ;                    RETAINED LOCK OF I/O DATABASE BEFORE CALL TO IOC$CREATE_UC
                        0000     43 ;                    AS THAT IS THE NEW INTERFACE TO IOC$CREATE_UCB.
                        0000     44 ;
                        0000     45 ;       V04          ADE0002         A.D.E           22-JUN-1979     13:00
                        0000     46 ;                    Test UCB$V_TEMPLATE in UCB$W_STS to determine if a
                        0000     47 ;                    UCB needs to be created rather than deadreckoning
                        0000     48 ;                    on UCB$NET0
                        0000     49 ;
                        0000     50 ;       V03          SGD0001         S.G.D.          04-APR-1979     17:10
                        0000     51 ;                    Use "_NET" instead of "NET".
```

```
                                0000     52 ;
                                0000     53 ;       V02     TGD0001         T.G. DOPIRAK    22-FEB-1979
                                0000     54 ;               RETURN SS$_DEVACTIVE IF MAILBOX ALREADY ASSOCIATED
                                0000     55 ;               WITH DEVICE.
                                0000     56 ;
                                0000     57 ;--
                                0000     58
                                0000     59 ;
                                0000     60 ; MACRO LIBRARY CALLS
                                0000     61 ;
                                0000     62
                                0000     63         $CCBDEF                          ;DEFINE CCB OFFSETS
                                0000     64         $IODEF                           ;DEFINE I/O FUNCTION CODES
                                0000     65         $JIBDEF                          ;DEFINE JIB OFFSETS
                                0000     66         $LOGDEF                          ;DEFINE LOG OFFSETS
                                0000     67         $PCBDEF                          ;DEFINE PCB OFFSETS
                                0000     68         $PRDEF                           ;DEFINE PROCESSOR REGISTERS
                                0000     69         $PRVDEF                          ;DEFINE PRIVILEGE BITS
                                0000     70         $SSDEF                           ;DEFINE SYSTEM STATUS VALUES
                                0000     71         $UCBDEF                          ;DEFINE UCB OFFSETS
                                0000     72
                                0000     73 ;
                                0000     74 ; LOCAL SYMBOLS
                                0000     75 ;
                                0000     76 ; ARGUMENT LIST OFFSET DEFINITIONS
                                0000     77 ;
                                0000     78
                    00000004    0000     79 DEVNAM=4                                 ;ADDRESS OF DEVICE NAME STRING DESCRIPTOR
                    00000008    0000     80 CHAN=8                                   ;ADDRESS TO STORE ASSIGNED CHANNEL NUMBER
                    0000000C    0000     81 ACMODE=12                                ;ACCESS MODE
                    00000010    0000     82 MBXNAM=16                                ;ADDRESS OF MAILBOX NAME STRING DESCRIPTOR
                                0000     83
                                0000     84 ;
                                0000     85 ; LOCAL DATA
                                0000     86 ;
                                0000     87
V54                         00000000     .1          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
-1                  54 45 4E 5F  0000     89 NETNAM:  .ASCII  /_NET/                  ;NETWORK DEVICE NAME
                                0004     90 NETEND:                                  ;REFERENCE LABEL
```

```
                        0004     92          .SBTTL  ASSIGN I/O CHANNEL
                        0004     93 ;+
                        0004     94 ; EXE$ASSIGN - ASSIGN I/O CHANNEL
                        0004     95 ;
                        0004     96 ; THIS SERVICE PROVIDES THE CAPABILITY TO ASSIGN A DEVICE TO AN I/O CHANNEL
                        0004     97 ; AND ESTABLISH NECESSARY DEVICE LINKAGE AND CONTROL INFORMATION IN THE
                        0004     98 ; ASSOCIATED CHANNEL CONTROL BLOCK. OPTIONALLY A MAILBOX CAN ALSO BE
                        0004     99 ; SPECIFIED WHICH WILL RECEIVE UNSOLICITED INPUT SENT TO THE ASSIGNED
                        0004    100 ; DEVICE.
                        0004    101 ;
                        0004    102 ; INPUTS:
                        0004    103 ;
                        0004    104 ;       DEVNAM(AP) = ADDRESS OF DEVICE NAME STRING DESCRIPTOR.
                        0004    105 ;       CHAN(AP) = ADDRESS TO STORE ASSIGNED CHANNEL NUMBER.
                        0004    106 ;       ACMODE(AP) = ACCESS MODE CHANNEL IS TO BE ASSIGNED TO.
                        0004    107 ;       MBXNAM(AP) = ADDRESS OF MAILBOX NAME STRING DESCRIPTOR (ZERO IMPLIES
                        0004    108 ;               NONE).
                        0004    109 ;
                        0004    110 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        0004    111 ;
                        0004    112 ; OUTPUTS:
                        0004    113 ;
                        0004    114 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO ASSIGN CHANNEL TO DEVICE.
                        0004    115 ;
                        0004    116 ;               R0 = SS$_ACCVIO - DEVICE NAME STRING, DEVICE NAME STRING
                        0004    117 ;                       DESCRIPTOR, MAILBOX NAME STRING, OR MAILBOX NAME
                        0004    118 ;                       STRING DESCRIPTOR CANNOT BE READ BY CALLING ACCESS
                        0004    119 ;                       MODE, OR CHANNEL NUMBER CANNOT BE WRITTEN BY CALLING
                        0004    120 ;                       ACCESS MODE.
                        0004    121 ;
                        0004    122 ;       R0 = SS$_DEVALLOC - DEVICE ALLOCATED TO ANOTHER PROCESS.
                        0004    123 ;
                        0004    124 ;       R0 = SS$_DEVNOTMBX - SPECIFIED MAILBOX DEVICE IS NOT A
                        0004    125 ;               MAILBOX.
                        0004    126 ;
                        0004    127 ;       R0 = SS$_EXQUOTA - PROCESS HAS INSUFFICIENT BUFFER QUOTA TO
                        0004    128 ;               ALLOCATE NETWORK UCB.
                        0004    129 ;
                        0004    130 ;       R0 = SS$_INSFMEM - SUFFICIENT SYSTEM DYNAMIC MEMORY DOES NOT
                        0004    131 ;               EXIST TO ALLOCATE NETWORK UCB.
                        0004    132 ;
                        0004    133 ;       R0 = SS$_IVDEVNAM - DEVICE OR MAILBOX NAME STRING CONTAINS
                        0004    134 ;               INVALID CHARACTERS, OR NO DEVICE NAME STRING
                        0004    135 ;               DESCRIPTOR SPECIFIED.
                        0004    136 ;
                        0004    137 ;       R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
                        0004    138 ;               DEVICE OR MAILBOX NAME STRING SPECIFIED.
                        0004    139 ;
                        0004    140 ;       R0 = SS$_NOIOCHAN - NO I/O CHANNEL IS AVAILABLE FOR ASSIGNMENT.
                        0004    141 ;
                        0004    142 ;       R0 = SS$_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO CREATE
                        0004    143 ;               NETWORK UCB OR DOES NOT HAVE PRIVILEGE TO ALLOCATE
                        0004    144 ;               THE DEVICE.
                        0004    145 ;
                        0004    146 ;       R0 = SS$_NOSUCHDEV - SPECIFIED DEVICE OR MAILBOX DOES NOT
                        0004    147 ;               EXIST ON HOST SYSTEM.
                        0004    148 ;
```

```
                                0004   149 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0004   150 ;
                                0004   151 ;               R0 = SS$_REMOTE - NORMAL COMPLETION, ASSIGNMENT COMPLETED
                                0004   152 ;                    ON REMOTE SYSTEM.
                                0004   153 ;
                                0004   154 ;               R0 = SS$_NORMAL - NORMAL COMPLETION, ASSIGNMENT COMPLETED
                                0004   155 ;                    ON HOST SYSTEM.
                                0004   156 ;
                                0004   157 ;               R0 = SS$_DEVACTIVE - MAILBOX ALREADY ASSOCIATED WITH DEVICE
                                0004   158 ;-
                                0004   159
                          0FFC  0004   160          .ENTRY  EXE$ASSIGN,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
V54             00000000'EF    9E  0006    .1      MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
                           54       000C
                 5B    08 AC    D0  000D   161      MOVL    CHAN(AP),R11            ;GET ADDRESS TO STORE CHANNEL NUMBER
                                    0011   162      IFNOWRT #2,(R11),30$            ;CAN CHANNEL NUMBER BE WRITTEN?
                              6B    B4  0017   163      CLRW    (R11)                   ;CLEAR CHANNEL NUMBER IN CASE OF ERROR
                 5A    10 AC    D0  0019   164      MOVL    MBXNAM(AP),R10          ;GET ADDRESS OF MAILBOX NAME DESCRIPTOR
                              0C    13  001D   165      BEQL    10$                     ;IF EQL NO MAILBOX SPECIFIED
                                    001F   166      IFNORD  #8,(R10),30$            ;CAN MAILBOX DESCRIPTOR BE READ?
                     7E    6A    7D  0025   167      MOVQ    (R10),-(SP)             ;COPY MAILBOX NAME DESCRIPTOR
                     5A    5E    D0  0028   168      MOVL    SP,R10                  ;SET ADDRESS OF MAILBOX NAME DESCRIPTOR
               50    0144  8F    3C  002B   169 10$:  MOVZWL  #SS$_IVDEVNAM,R0        ;SET INVALID DEVICE NAME STATUS
                 59    04 AC    D0  0030   170      MOVL    DEVNAM(AP),R9           ;GET ADDRESS OF DEVICE NAME DESCRIPTOR
                              18    13  0034   171      BEQL    20$                     ;IF EQL NO DEVICE SPECIFIED
                                    0036   172      IFNORD  #8,(R9),30$             ;CAN DEVICE NAME DESCRIPTOR BE READ?
                     02    00    EF  003C   173      EXTZV   #0,#2,ACMODE(AP),R0     ;GET SPECIFIED ACCESS MODE
               50    0C AC          003F
                        FF88'  30    0042   174      BSBW    EXE$MAXACMODE           ;MAXIMIZE ACCESS MODE
                 58    50    D0  0045   175      MOVL    R0,R8                   ;SAVE MAXIMIZED ACCESS MODE
                        FFB5'  30    0048   176      BSBW    IOC$FFCHAN              ;FIND FREE I/O CHANNEL
                     05 50    E8  004B   177      BLBS    R0,FREECHAN             ;IF LBS FREE I/O CHANNEL FOUND
                              04    004E   178 20$:  RET                             ;
                 50    0C    3C  004F   179 30$:  MOVZWL  #SS$_ACCVIO,R0           ;SET ACCESS VIOLATION STATUS
                              04    0052   180      RET                             ;
                                    0053   181
                                    0053   182 ;
                                    0053   183 ; FREE CHANNEL FOUND
                                    0053   184 ;
                                    0053   185
                                    0053   186          .ENABL  LSB
                                    0053   187 FREECHAN:                            ;FREE CHANNEL FOUND
                 57    51    D0  0053   188          MOVL    R1,R7                   ;SAVE CHANNEL INDEX
-1                      0082  30    0056   190          BSBW    TEST_MAILBOX            ;TEST IF MAILBOX SPECIFIED
                     3D 50    E9  0059   191          BLBC    R0,30$                  ;IF LBC SEARCH FAILURE
                 51    59    D0  005C   192          MOVL    R9,R1                   ;SET ADDRESS OF DEVICE NAME DESCRIPTOR
                        FF9E'  30    005F   193          BSBW    IOC$SEARCHDEV           ;SEARCH FOR DEVICE
                     36 50    E9  0062   194          BLBC    R0,40$                  ;IF LBC SEARCH FAILURE
                                    0065   195
                                    0065   196 ;
                                    0065   197 ; DEVICE FOUND
                                    0065   198 ;
                                    0065   199
                 55    51    D0  0065   200          MOVL    R1,R5                   ;SAVE ADDRESS OF DEVICE UCB
             0C 34 A5    00'   E0  0068   201          BBS     S^#DEV$V_SPL,UCB$L_DEVCHAR(R5),80$ ;IF SET, SPOOLED DEVICE
                              0D    E1  006D   202          BBC     S^#UCB$V_TEMPLATE,-     ;IF CLR, ASSIGNING TO A LOCAL DEVICE
                    03 58 A5          006F   203                  UCB$W_STS(R5),LOCAL     ;
```

```
                 0084   31  0072   204          BRW     NETWORK                  ;IF SET, NETWORK ASSIGN
                        0075   205
                        0075   206 ;
                        0075   207 ; LOCAL ASSIGNMENT
                        0075   208 ;
                        0075   209
                        0075   210 LOCAL:                                        ;LOCAL ASSIGNMENT
          50    28 A5   D0   0075   211          MOVL    UCB$L_PID(R5),R0         ;GET PROCESS ID OF OWNER
                2A      13   0079   212          BEQL    50$                      ;IF EQL DEVICE NOT ALLOCATED
             51   54    D0   007B   213          MOVL    R4,R1                    ;COPY PROCESS PCB ADDRESS
          60 A1   50    D1   007E   214 10$:     CMPL    R0,PCB$L_PID(R1)         ;PROCESS ID MATCH?
                2B      13   0082   215          BEQL    70$                      ;IF EQL YES
          51    1C A1   3C   0084   216          MOVZWL  PCB$L_OWNER(R1),R1       ;GET CREATOR PROCESS INDEX
                0A      13   0088   217          BEQL    20$                      ;IF EQL NO CREATOR
     00000000'FF41   D0   008A   218          MOVL    @L^SCH$GL_PCBVEC[R1],R1  ;GET ADDRESS OF CREATOR PCB
                51           0091
                EA      11   0092   219          BRB     10$                      ;
          50    0840 8F   3C   0094   220 20$:     MOVZWL  #SS$_DEVALLOC,R0         ;SET DEVICE ALREADY ALLOCATED
                5B      11   0099   221 30$:     BRB     90$                      ;
                        009B   222
                        009B   223 ;
                        009B   224 ; DEVICE SEARCH FAILURE
                        009B   225 ;
                        009B   226
          50    08F0 8F   B1   009B   227 40$:     CMPW    #SS$_NONLOCAL,R0         ;REMOTE DEVICE?
                  54      12   00A0   228          BNEQ    90$                      ;IF NEQ NO
                005D      31   00A2   229          BRW     REMOTE                   ;
                        00A5   230
                        00A5   231 ;
                        00A5   232 ; DEVICE NOT SPOOLED OR ALLOCATED - IF IT'S ALSO NOT SHAREABLE, CHECK THAT
                        00A5   233 ; PROCESS HAS PRIVILEGE TO ALLOCATE IT
                        00A5   234 ;
                        00A5   235
    05 34 A5    00'  E0   00A5   236 50$:     BBS     S^#DEV$V_SHR,UCB$L_DEVCHAR(R5),70$ ;IF SET, DEVICE SHAREABLE
-5                      00AA   242 ;
                        00AA   243 ; NONSHAREABLE DEVICE - PERFORM IMPLICIT ALLOCATION
                        00AA   244 ;
                        00AA   245
    28 A5    60 A4   D0   00AA   246 60$:     MOVL    PCB$L_PID(R4),UCB$L_PID(R5) ;SET CURRENT PROCESS AS OWNER
                        00AF   247
                        00AF   248 ;
                        00AF   249 ; ASSOCIATE MAILBOX IF:
                        00AF   250 ;
                        00AF   251 ;       1. NOT FILE DEVICE
                        00AF   252 ;       2. NOT SHAREABLE DEVICE
                        00AF   253 ;       3. MAILBOX NOT ALREADY ASSOCIATED
                        00AF   254 ;       4. MAILBOX IS SPECIFIED
                        00AF   255 ;
                        00AF   256
    25 34 A5    00'  E0   00AF   257 70$:     BBS     S^#DEV$V_FOD,UCB$L_DEVCHAR(R5),80$ ;IF SET, FILE DEVICE
    20 34 A5    00'  E0   0084   258          BBS     S^#DEV$V_SHR,UCB$L_DEVCHAR(R5),80$ ;IF SET, SHARED DEVICE
                56   D5   0089   259          TSTL    R6                       ;ARE WE ASSOCIATING A MBX
                1C   13   008B   260          BEQL    80$                      ;IF NOT JUST CONTINUE
             54 A5   D5   008D   261          TSTL    UCB$L_AMB(R5)            ;IS THERE ONE CURRENTLY ASSOC?
                0D   13   00C0   262          BEQL    75$                      ;IF NOT ASSOC NEW ONE
          54 A5   56   D1   00C2   263          CMPL    R6,UCB$L_AMB(R5)         ;TRYING TO ASSOC DIFFERENT MBX?
                11   13   00C6   264          BEQL    80$                      ;IF NOT JUST CONTINUE
```

```
                50   02C4 8F   3C  00C8   265          MOVZWL  #SS$_DEVACTIVE,R0       ;DON'T DO THE ASSIGN
                           27   11  00CD   266          BRB     90$                     ;RETURN THE ERROR
                                    00CF   267 75$:
                  54 A5    56   D0  00CF   268          MOVL    R6,UCB$L_AMB(R5)        ;SET ASSOCIATED MAILBOX UCB ADDRESS
                     50 A6      B6  00D3   269          INCW    UCB$W_REFC(R6)          ;INCREMENT MAILBOX UCB REFERENCE COUNT
                     56   01    9A  00D6   270          MOVZBL  #CCB$M_AMB,R6           ;SET ASSOCIATED MAILBOX FLAG
               00000000'FF47    9E  00D9   271 80$:     MOVAB   @CTL$GL_CCBBASE[R7],R0  ;COMPUTE BASE OF CCB
                           50       00E0
                  60   55    D0  00E1   272          MOVL    R5,CCB$L_UCB(R0)        ;STORE UCB ADDRESS IN CCB
                     50 A5      B6  00E4   273          INCW    UCB$W_REFC(R5)          ;INCREMENT UCB REFERENCE COUNT
                  58   01    81  00E7   274          ADDB3   #1,R8,CCB$B_AMOD(R0)    ;STORE ACCESS MODE OF CHANNEL
                     09 A0       00EA
               08 A0   56    90  00EC   275          MOVB    R6,CCB$B_STS(R0)        ;SET CHANNEL STATUS FLAGS
                  6B   57    AE  00F0   276          MNEGW   R7,(R11)                ;STORE ASSIGNED CHANNEL NUMBER
                  50   01    3C  00F3   277          MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION STATUS
                       FF07'  31  00F6   278 90$:     BRW     IOC$UNLOCK              ;UNLOCK I/O DATA BASE AND RETURN
                                    00F9   279
                                    00F9   280 ;
                                    00F9   281 ; NETWORK ASSIGNMENT
                                    00F9   282 ;
                                    00F9   283
                                    00F9   284 NETWORK:                             ;
V54               50   08F0 8F   3C  00F9    .1          MOVZWL  #SS$_NONLOCAL,R0        ; Set return to non-local
V54                        FEFF'  30  00FE    .2          BSBW    IOC$UNLOCK              ; Unlock I/O database and return
V54                               04  0101    .3          RET
-27                               0102   312          .DSABL  LSB
                                  0102   313
                                  0102   314 ;
                                  0102   315 ; REMOTE DEVICE SPECIFIED
                                  0102   316 ;
                                  0102   317
                                  0102   318 REMOTE:                              ;
V54               50   08F0 8F   3C  0102    .1          MOVZWL  #SS$_NONLOCAL,R0        ; Set return to non-local
V54                        FEF6'  30  0107    .2          BSBW    IOC$UNLOCK              ; unlock i/o data base
V54                               04  010A    .3          RET
```

```
                                010B   356          .SBTTL  TEST IF MAILBOX SPECIFIED
                                010B   357 ;
                                010B   358 ; SUBROUTINE TO TEST IF A MAILBOX IS SPECIFIED
                                010B   359 ;
                                010B   360 ; INPUTS:
                                010B   361 ;
                                010B   362 ;       R10 = ADDRESS OF MAILBOX NAME DESCRIPTOR
                                010B   363 ;
                                010B   364 ; OUTPUTS:
                                010B   365 ;
                                010B   366 ;       R0 = SS$_NORMAL IF SPECIFIED MAILBOX EXISTS
                                010B   367 ;            SS$_NOSUCHDEV IF SPECIFIED MAILBOX DOES NOT EXIST
                                010B   368 ;            SS$_DEVNOTMBX IF SPECIFIED DEVICE IS NOT A MAILBOX
                                010B   369 ;       R6 = ADDRESS OF MAILBOX UCB
                                010B   370 ;            ZERO IF MAILBOX NOT SPECIFIED (USED AS CHANNEL STATUS FLAGS)
                                010B   371 ;
                                010B   372
                                010B   373 TEST_MAILBOX:                                ;
          56    5A    D0        010B   374          MOVL    R10,R6                      ;SET ADDRESS OF MAILBOX NAME DESCRIPTOR
                18    13        010E   375          BEQL    10$                         ;IF EQL NO NAME SPECIFIED
          51    56    D0        0110   376          MOVL    R6,R1                       ;COPY ADDRESS OF MAILBOX NAME DESCRIPTOR
             FEEA'    30        0113   377          BSBW    IOC$SEARCHDEV               ;SEARCH FOR DEVICE
            15 50     E9        0116   378          BLBC    R0,20$                      ;IF LBC SEARCH ERROR
     50     0074 8F   3C        0119   379          MOVZWL  #SS$_DEVNOTMBX,R0           ;SET DEVICE NOT MAILBOX STATUS
  0B 34 A1    00'    E1        011E   380          BBC     S^#DEV$V_MBX,UCB$L_DEVCHAR(R1),20$ ;IF CLR, DEVICE NOT MAILBOX
  06 34 A1    00'    F0        0123   381          BBS     S^#DEV$V_NET,UCB$L_DEVCHAR(R1),20$ ;IF SET, NETWORK DEVICE
          56    51    D0        0128   382          MOVL    R1,R6                       ;SAVE ADDRESS OF MAILBOX UCB
          50    01    3C        012B   383 10$:     MOVZWL  #SS$_NORMAL,R0              ;SET NORMAL COMPLETION STATUS
                      05        012E   384 20$:     RSB                                 ;
                                012F   385
                                012F   386          .END
```

```
ACMODE                          =  0000000C     D
CCB$B_AMOD                        00000009     D
CCB$B_STS                         00000008     D
CCB$C_LENGTH                      00000010     D
CCB$K_LENGTH                      00000010     D
CCB$L_DIRP                        0000000C     D
CCB$L_UCB                         00000000     D
CCB$L_WIND                        00000004     D
CCB$M_AMB                       =  00000001     D
CCB$W_IOC                         0000000A     D
CHAN                            =  00000008     D
CTL$GL_CCBBASE                    ********     X    02
DEV$V_FOD                         ********     X    02
DEV$V_MBX                         ********     X    02
DEV$V_NET                         ********     X    02
DEV$V_SHR                         ********     X    02
DEV$V_SPL                         ********     X    02
DEVNAM                          =  00000004     D
DS$AX_SOFTPCB                     ********     X    02
EXE$ASSIGN                        00000004 RG  D    02
EXE$MAXACMODE                     ********     X    02
FREECHAN                          00000053 R   D    02
IOC$FFCHAN                        ********     X    02
IOC$SEARCHDEV                     ********     X    02
IOC$UNLOCK                        ********     X    02
LOCAL                             00000075 R   D    02
MBXNAM                          =  00000010     D
NETEND                            00000004 R   D    02
NETNAM                            00000000 R   D    02
NETWORK                           00000F9C R   D    02
PCB$B_ASTACT                      0000000C     D
PCB$B_ASTEN                       0000000D     D
PCB$B_PRI                         0000000B     D
PCB$B_PRIB                        0000002F     D
PCB$B_TYPE                        0000000A     D
PCB$B_WFFC                        0000002E     D
PCB$C_LENGTH                      0000008C     D
PCB$K_LENGTH                      0000008C     D
PCB$L_ARB                         00000084     D
PCB$L_ASTQBL                      00000014     D
PCB$L_ASTQFL                      00000010     D
PCB$L_EFC2P                       00000058     D
PCB$L_EFC3P                       0000005C     D
PCB$L_EFCS                        00000050     D
PCB$L_EFCU                        00000054     D
PCB$L_EFWM                        0000004C     D
PCB$L_JIB                         00000078     D
PCB$L_OWNER                       0000001C     D
PCB$L_PHD                         00000064     D
PCB$L_PHYPCB                      00000013     D
PCB$L_PID                         00000060     D
PCB$L_PQB                         0000004C     D
PCB$L_SQBL                        00000004     D
PCB$L_SQFL                        00000000     D
PCB$L_STS                         00000024     D
PCB$L_UIC                         00000088     D
PCB$L_WSSWP                       00000029     D
PCB$L_WTIME                       00000028     D
PCB$Q_PRIV                        0000007C     D
PCB$T_LNAME                       00000068     D
PCB$T_TERMINAL                    00000044     D
PCB$W_APTCNT                      00000030     D
PCB$W_ASTCNT                      00000038     D
PCB$W_BIOCNT                      0000003A     D
PCB$W_BIOLM                       0000003C     D
PCB$W_DIOCNT                      0000003E     D
PCB$W_DIOLM                       00000040     D
PCB$W_GPGCNT                      00000034     D
PCB$W_GRP                         0000008A     D
PCB$W_MEM                         00000088     D
PCB$W_MTXCNT                      0000000E     D
PCB$W_PPGCNT                      00000036     D
PCB$W_PRCCNT                      00000042     D
PCB$W_SIZE                        00000008     D
PCB$W_STATE                       0000002C     D
PCB$W_TMBU                        00000032     D
REMOTE                            00000102 R   D    02
SCH$GL_PCBVEC                     ********     X    02
SIZ...                          =  00000002     D
SS$_ACCVIO                      =  0000000C     D
SS$_DEVACTIVE                   =  000002C4     D
SS$_DEVALLOC                    =  00000840     D
SS$_DEVNOTMBX                   =  00000074     D
SS$_IVDEVNAM                    =  00000144     D
SS$_NONLOCAL                    =  000008F0     D
SS$_NORMAL                      =  00000001     D
TEST_MAILBOX                      0000010B R   D    02
UCB$B_AMOD                        00000053     D
UCB$B_CEX                         00000077     D
UCB$B_CM1                         0000004A     D
UCB$B_CM2                         0000004B     D
UCB$B_DEVCLASS                    00000038     D
UCB$B_DEVTYPE                     00000039     D
UCB$B_DIPL                        00000052     D
UCB$B_DX_SCTCNT                   000000A6     D
UCB$B_ERTCNT                      00000070     D
UCB$B_ERTMAX                      00000071     D
UCB$B_FFX                         00000076     D
UCB$B_FIPL                        0000000B     D
UCB$B_LOCSRV                      0000003C     D
UCB$B_OFFNDX                      00000094     D
UCB$B_OFFRTC                      00000095     D
UCB$B_REMSRV                      0000003D     D
UCB$B_SECTORS                     0000003C     D
UCB$B_SLAVE                       00000074     D
UCB$B_SPR                         00000075     D
UCB$B_STATE                       00000052     D
UCB$B_TRACKS                      0000003D     D
UCB$B_TT_CRFILL                   0000009D     D
UCB$B_TT_DECRF                    000000A1     D
UCB$B_TT_DELFF                    000000A2     D
UCB$B_TT_DESPEE                   000000A0     D
UCB$B_TT_DETYPE                   000000A4     D
UCB$B_TT_LFFILL                   0000009E     D
```

UCB$B_TT_SPEED	0000009C	D	UCB$V_TEMPLATE	=	0000000D	D
UCB$B_TYPE	0000000A	D	UCB$W_BCNT		0000006C	D
UCB$B_VERTSZ	0000003F	D	UCB$W_BCR		00000096	D
UCB$C_LENGTH	00000074	D	UCB$W_BOFF		0000006C	D
UCB$C_MB_LENGTH	00000090	D	UCB$W_BUFQUO		00000018	D
UCB$C_TT_LENGTH	000000BC	D	UCB$W_BYTESTOGO		0000003E	D
UCB$K_LENGTH	00000074	D	UCB$W_CHARGE		0000004A	D
UCB$K_MB_LENGTH	00000090	D	UCB$W_CYLINDERS		0000003E	D
UCB$K_TT_LENGTH	000000BC	D	UCB$W_DA		0000008C	D
UCB$L_AMB	00000054	D	UCB$W_DC		0000008E	D
UCB$L_ASTQBL	00000010	D	UCB$W_DEVBUFSIZ		0000003A	D
UCB$L_ASTQFL	0000000C	D	UCB$W_DEVSTS		0000005A	D
UCB$L_CPID	0000005C	D	UCB$W_DIRSEQ		00000088	D
UCB$L_CRB	00000020	D	UCB$W_DSTADDR		00000018	D
UCB$L_DDB	0000C024	D	UCB$W_DX_BCR		000000A4	D
UCB$L_DEVCHAR	00000034	D	UCB$W_ECT		00000090	D
UCB$L_DEVDEPEND	0000003C	D	UCB$W_EC2		00000092	D
UCB$L_DPC	00000080	D	UCB$W_ERRCNT		00000072	D
UCB$L_DUETIM	0000005C	D	UCB$W_FUNC		0000007E	D
UCB$L_DX_BFPNT	0000009C	D	UCB$W_MB_SEED		00000000	D
UCB$L_DX_BUF	00000098	D	UCB$W_MSGCNT		00000016	D
UCB$L_DX_RXDB	000000A0	D	UCB$W_MSGMAX		00000014	D
UCB$L_EMB	00000078	D	UCB$W_NT_CHAN		0000007C	D
UCB$L_FIRST	00000014	D	UCB$W_OFFSET		0000008A	D
UCB$L_FPC	0000000C	D	UCB$W_REFC		00000050	D
UCB$L_FQBL	00000004	D	UCB$W_SIZE		00000008	D
UCB$L_FQFL	00000000	D	UCB$W_SRCADDR		0000001A	D
UCB$L_FR3	00000010	D	UCB$W_STS		00000058	D
UCB$L_FR4	00000014	D	UCB$W_TT_DESIZE		000000A5	D
UCB$L_IOQBL	00000044	D	UCB$W_UNIT		00000048	D
UCB$L_IOQFL	00000040	D	UCB$W_VPROT		0000001A	D
UCB$L_IRP	0000004C	D				
UCB$L_LINK	0000002C	D				
UCB$L_LOGADR	00000064	D				
UCB$L_MAXBLOCK	00000084	D				
UCB$L_MB_MBX	0000007C	D				
UCB$L_MB_PORT	0000008C	D				
UCB$L_MB_RAST	00000078	D				
UCB$L_MB_SHB	00000080	D				
UCB$L_MB_WAST	00000074	D				
UCB$L_MB_WIOQBL	00000088	D				
UCB$L_MB_WIOQFL	00000084	D				
UCB$L_MEDIA	0000008C	D				
UCB$L_NT_DATSSB	00000074	D				
UCB$L_NT_INTSSB	00000078	D				
UCB$L_OPCNT	00000060	D				
UCB$L_OWNUIC	0000001C	D				
UCB$L_PID	00000028	D				
UCB$L_RQBL	00000004	D				
UCB$L_RQFL	00000000	D				
UCB$L_SVAPTE	00000068	D				
UCB$L_SVPN	00000064	D				
UCB$L_TT_DECHAR	000000A8	D				
UCB$L_TT_RDUE	0000008C	D				
UCB$L_TT_RTIMOU	00000088	D				
UCB$L_VCB	00000030	D				
UCB$T_PARTNER	0000000C	D				

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	0000012F (303.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACMODE	=0000000C	81 (1)	173 (1)
CCB$B_AMOD	00000009		#-274 (1)
CCB$B_STS	00000008		#-275 (1)
CCB$L_UCB	00000000		#-272 (1)
CCB$M_AMB	=00000001		#-270 (1)
CHAN	=00000008	80 (1)	#-161 (1)
CTL$GL_CCBBASE	00000000-XR		271 (1)
DEV$V_FOD	00000C00-XR		#-257 (1)
DEV$V_MBX	00000000-XR		#-380 (1)
DEV$V_NET	00000000-XR		#-381 (1)
DEV$V_SHR	00000000-XR		#-236 (1) #-258 (1)
DEV$V_SPL	00000000-XR		#-201 (1)
DEVNAM	=00000004	79 (1)	#-170 (1)
DS$AX_SOFTPCB	00000000-XR		160.1 (1)
EXE$ASSIGN	00000004-R	160 (1)	
EXE$MAXACMODE	00000000-XR		#-174 (1)
FREECHAN	00000053-R	187 (1)	#-177 (1)
IOC$FFCHAN	00000000-XR		#-176 (1)
IOC$SEARCHDEV	00000000-XR		#-193 (1) #-377 (1)
IOC$UNLOCK	00000000-XR		#-278 (1) #-284.2 (1) #-318.2 (1)
LOCAL	00000075-R	210 (1)	#-203 (1)
MBXNAM	=00000010	82 (1)	#-164 (1)
NETEND	00000004-R	90 (1)	
NETNAM	00000000-R	89 (1)	
NETWORK	000000F9-R	284 (1)	#-204 (1)
PCB$L_OWNER	0000001C		#-216 (1)
PCB$L_PID	00000060		#-214 (1) #-246 (1)
REMOTE	00000102-R	318 (1)	#-229 (1)
SCH$GL_PCBVEC	00000000-XR		#-218 (1)
SS$_ACCVIO	=0000000C		#-179 (1)
SS$_DEVACTIVE	=000002C4		#-265 (1)
SS$_DEVALLOC	=00000840		#-220 (1)
SS$_DEVNOTMBX	=00000074		#-379 (1)
SS$_IVDEVNAM	=00000144		#-169 (1)
SS$_NONLOCAL	=000008F0		#-227 (1) #-284.1 (1) #-318.1 (1)
SS$_NORMAL	=00000001		#-277 (1) #-383 (1)
TEST_MAILBOX	0000010B-R	373 (1)	#-190 (1)
UCB$L_AMB	00000054		#-261 (1) #-263 (1) #-268 (1)
UCB$L_DEVCHAR	00000034		201 (1) 236 (1) 257 (1) 258 (1)
			380 (1) 381 (1)
UCB$L_PID	00000028		#-211 (1) #-246 (1)
UCB$V_TEMPLATE	=0000000D		#-202 (1)
UCB$W_REFC	00000050		#-269 (1) #-273 (1)
UCB$W_STS	00000058		203 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CCBDEF	1	63 (1)	63 (1)
$DEFINI	1	63 (1)	63 (1) 64 (1) 65 (1) 66 (1) 67 (1) 68 (1) 69 (1) 70 (1) 71 (1)
$IODEF	17	64 (1)	64 (1)
$JIBDEF	3	65 (1)	65 (1)
$LOGDEF	1	66 (1)	66 (1)
$PCBDEF	4	67 (1)	67 (1)
$PRDEF	4	68 (1)	68 (1)
$PRVDEF	4	69 (1)	69 (1)
$SSDEF	21	70 (1)	70 (1)
$UCBDEF	10	71 (1)	71 (1)
IFNORD	1	166 (1)	166 (1) 172 (1)
IFNOWRT	1	162 (1)	162 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.10	00:00:00.46
Command processing	145	00:00:00.73	00:00:01.72
Pass 1	738	00:00:14.94	00:00:21.25
Symbol table sort	0	00:00:02.00	00:00:02.92
Pass 2	87	00:00:02.53	00:00:05.12
Symbol table output	25	00:00:00.17	00:00:00.38
Psect synopsis output	6	00:00:00.02	00:00:00.39
Cross-reference output	24	00:00:00.23	00:00:00.40
Assembler run totals	1063	00:00:20.76	00:00:32.67

The working set limit was 1000 pages.
72168 bytes (141 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1270 non-local and 15 local symbols.
330 source lines were read in Pass 1, producing 0 object records in Pass 2.
57 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	16

1527 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ASSIGN/UPDA=(ASSIGN.UPD,ASSIGN.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 40 HISTORY ; DETAILED
(1) 58 DECLARATIONS
(1) 87 FXE$SETAST - SET AST ENABLES

```
0000     1          .TITLE  ASTCON  *** ASTCON AST control service
0000     2          .IDENT  '07-02'
0000     3          .IDENT  'V02-001'               ;
0000     4
0000     5  ;
0000     6  ;*****************************************************************************
0000     7  ;*                                                                           *
0000     8  ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
0000     9  ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED    *
0000    12  ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
0000    14  ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS    *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26
0000    27  ;++
0000    28  ; FACILITY: EXECUTIVE, SYSTEM SERVICES
0000    29  ;
0000    30  ; ABSTRACT:
0000    31  ;       THIS MODULE CONTAINS THE SYSTEM SERVICES SETAST WHICH CONTROLS
0000    32  ;       THE STATE OF THE AST ENABLES AND DCLAST WHICH DECLARES AN
0000    33  ;       AST FOR A PARTICULAR ACCESS MODE.
0000    34  ;
0000    35  ; ENVIRONMENT:
0000    36  ;
0000    37  ;--
0000    38  ;
0000    39  ;       .PAGE
0000    40  ;       .SBTTL  HISTORY                 ; DETAILED
0000    41  ;
0000    42  ; AUTHOR:       R. HUSTVEDT      CREATION DATE: 21-SEP-76
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;       , : VERSION
0000    46  ; 01     -
0000    47  ;
0000    48  ;               Dave Butenhof   18-jun-1980,    version 5.5
0000    49  ;       01      Use VMS SYSASTCON module to implement supervisor SETAST
0000    50  ;               service.
0000    51  ;
0000    .1  ;       02      - Jack Stansbury, 23-Oct-1981, Version 6.-
-1      0000    53  ;               Fixed some truncation errors. Added .LIBRARY statements
0000    54  ;               for $DS and $DIAG. Removed previous .PSECT SEP statement,
0000    55  ;               changed it to .PSECT CODE.
0000    56
```

```
                        0000     58          .SBTTL  DECLARATIONS
                        0000     59
                        0000     60 ;
                        0000     61 ; INCLUDE FILES:
                        0000     62 ;
                        0000     63          .LIBRARY        /$DS/           ;                                       [02]
                        0000     64          .LIBRARY        /$DIAG/         ;                                       [02]
                        0000     65                                          ;                                       [02]
                        0000     66          $ACBDEF                         ; DEFINE AST CONTROL BLOCK OFFSETS
                        0000     67          $IPLDEF                         ; DEFINE IPL VALUES
                        0000     68          $PCBDEF                         ; DEFINE PCB OFFSETS
                        0000     69          $PRDEF                          ; DEFINE PROCESSOR REGISTER VALUES
                        0000     70          $PSLDEF                         ; DEFINE PSL FIELDS
                        0000     71          $RSNDEF                         ; DEFINE RESOURCE NUMBERS
                        0000     72          $SSDEF                          ; DEFINE STATUS CODE VALUES
                        0000     73
                        0000     74 ;
                        0000     75 ; EQUATES:
                        0000     76 ;
                        0000     77
0000000C                0000     78 ACMODE=12                               ; DISPLACEMENT TO ACCESS MODE
00000004                0000     79 AST=4                                   ; DISPLACEMENT TO AST ADDRESS
00000008                0000     80 ASTPRM=8                                ; DISPLACEMENT TO AST PARAMETER
00000004                0000     81 ENABLE=4                                ; DISPLACEMENT TO ENABLE ARGUMENT
                        0000     82 ;
                        0000     83 ; OWN STORAGE:
                        0000     84 ;
                    00000000     85          .PSECT  Y$EXEPAGED,BYTE        ; PAGED PSECT
```

```
                                0000     87          .SBTTL  EXE$SETAST - SET AST ENABLES
                           00000000     88          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                   [02]
                                0000     89
                                0000     90 ;++
                                0000     91 ; FUNCTIONAL DESCRIPTION:
                                0000     92 ;       EXE$SETAST ALLOWS A PROCESS TO ENABLE OR DISABLE THE DELIVERY
                                0000     93 ;       OF ASTS FOR THE PROCESS AT THE ACCESS MODE ISSUING THE CALL TO
                                0000     94 ;       EXE$SETAST.   THE PREVIOUS STATE OF THE AST ENABLE FOR THAT
                                0000     95 ;       ACCESS MODE IS RETURNED IN BIT 1 OF REGISTER 0 AS PART OF
                                0000     96 ;       THE SUCCESSFUL COMPLETION STATUS CODE.
                                0000     97 ;
                                0000     98 ; CALLING SEQUENCE:
                                0000     99 ;       CALLG   ARGLIST,EXE$SETAST
                                0000    100 ;
                                0000    101 ; INPUT PARAMETERS:
                                0000    102 ;       04(AP) - NEW SETTING FOR AST ENABLE
                                0000    103 ;       R4 - PCB ADDRESS OF CURRENT PROCESS
                                0000    104 ;
                                0000    105 ; IMPLICIT INPUTS:
                                0000    106 ;       SCH$GL_CURPCB - POINTER TO CURRENT PROCESS PCB
                                0000    107 ;
                                0000    108 ; OUTPUT PARAMETERS:
                                0000    109 ;       R0 - COMPLETION STATUS CODE
                                0000    110 ;
                                0000    111 ; IMPLICIT OUTPUTS:
                                0000    112 ;       AST ENABLE FOR PREVIOUS MODE IS SET ACCORDING TO ARGUMENT
                                0000    113 ;
                                0000    114 ;       THE NEW SETTING FOR THE ASTLVL REGISTER IS PLACED BOTH IN
                                0000    115 ;       THE PROCESSOR REGISTER AND THE CURRENT HARDWARE PCB.
                                0000    116 ;
                                0000    117 ; COMPLETION CODES:
                                0000    118 ;       SS$_WASCLR  -  AST ENABLE FOR ACCESS MODE WAS CLEAR
                                0000    119 ;       SS$_WASSET  -  AST ENABLE FOR ACCESS MODE WAS SET
                                0000    120 ;
                                0000    121 ; SIDE EFFECTS:
                                0000    122 ;       ANY PENDING ASTS FOR THIS PROCESS WHICH BECOME DELIVERABLE
                                0000    123 ;       AS A RESULT OF SETTING AN AST ENABLE MAY BE DELIVERED UPON
                                0000    124 ;       EXIT FROM THIS SERVICE.
                                0000    125 ;
                                0000    126 ;--
                                0000    127
                           003C 0000    128          .ENTRY  EXE$SETAST,^M<R2,R3,R4,R5>
  00000000'EF   DE             0002    129          MOVAL   DS$AX_SOFTPCB, R4           ; Get PCB address
          54                   0008
          51   DC              0009    130          MOVPSL  R1                          ; GET CURRENT PSL
51   02   18   EF              0008    131          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1 ; Extract current mode
          51                   000F
     55   09   3C              0010    132          MOVZWL  #SS$_WASSET,R5              ; ASSUME SET
03 0D A4   51   E0              0013    133          BBS     R1,PCB$B_ASTEN(R4),10$      ; TEST OLD ENABLE SETTING
     55   01   3C              0018    134          MOVZWL  #SS$_WASCLR,R5              ; IT WAS CLEAR
  51   04 AC   F0              001B    135 10$:     INSV    ENABLE(AP),R1,#1,PCB$B_ASTEN(R4)  ; SET NEW ENABLE VALUE
  0D A4   01                   001F
  00000000'EF   16             0022    136          JSB     SCH$NEWLVL                  ;:: COMPUTE NEW ASTLVL                  [02]
     50   55   D0              0028    137          MOVL    R5,R0                       ; SET COMPLETION STATUS
               04              0028    138          RET                                 ; AND RETURN TO CALLER
                               002C    139          .END
```

```
ACB$B_RMOD                      0000000B     D
ACB$B_TYPE                      0000000A     D
ACB$C_LENGTH                    00000018     D
ACB$K_LENGTH                    00000018     D
ACB$L_AST                       00000010     D
ACB$L_ASTPRM                    00000014     D
ACB$L_ASTQBL                    00000004     D
ACB$L_ASTQFL                    00000000     D
ACB$L_KAST                      00000018     D
ACB$L_PID                       0000000C     D
ACB$W_SIZE                      00000008     D
ACMODE                        = 0000000C     D
AST                           = 00000004     D
ASTPRM                        = 00000008     D
DS$AX_SOFTPCB                   ********   X   03
ENABLE                        = 00000004     D
EXE$SETAST                      00000000 RG  D   03
PCB$B_ASTACT                    0000000C     D
PCB$B_ASTEN                     0000000D     D
PCB$B_PRI                       0000000B     D
PCB$B_PRIB                      0000002F     D
PCB$B_TYPE                      0000000A     D
PCB$B_WEFC                      0000002E     D
PCB$C_LENGTH                    0000008C     D
PCB$K_LENGTH                    0000008C     D
PCB$L_ARB                       00000084     D
PCB$L_ASTQBL                    00000014     D
PCB$L_ASTQFL                    00000010     D
PCB$L_EFC2P                     00000058     D
PCB$L_EFC3P                     0000005C     D
PCB$L_EFCS                      00000050     D
PCB$L_EFCU                      00000054     D
PCB$L_EFWM                      0000004C     D
PCB$L_JIB                       00000078     D
PCB$L_OWNER                     0000001C     D
PCB$L_PHD                       00000064     D
PCB$L_PHYPCB                    00000018     D
PCB$L_PID                       00000060     D
PCB$L_PQB                       0000004C     D
PCB$L_SQBL                      00000004     D
PCB$L_SQFL                      00000000     D
PCB$L_STS                       00000024     D
PCB$L_UIC                       00000088     D
PCB$L_WSSWP                     00000020     D
PCB$L_WTIME                     00000028     D
PCB$Q_PRIV                      0000007C     D
PCB$T_LNAME                     00000068     D
PCB$T_TERMINAL                  00000044     D
PCB$W_APTCNT                    00000030     D
PCB$W_ASTCNT                    00000038     D
PCB$W_BIOCNT                    0000003A     D
PCB$W_BIOLM                     0000003C     D
PCB$W_DIOCNT                    0000003E     D
PCB$W_DIOLM                     00000040     D
PCB$W_GPGCNT                    00000034     D
PCB$W_GRP                       0000008A     D
PCB$W_MEM                       00000088     D
PCB$W_MTXCNT                    0000000E     D
PCB$W_PPGCNT                    00000036     D
PCB$W_PRCCNT                    00000042     D
PCB$W_SIZE                      00000008     D
PCB$W_STATE                     0000002C     D
PCB$W_TMBU                      00000032     D
PSL$S_CURMOD                  = 00000002     D
PSL$V_CURMOD                  = 00000018     D
SCH$NEWLVL                      ********   X   03
SIZ...                        = 00000001     D
SS$_WASCLR                    = 00000001     D
SS$_WASSET                    = 00000009     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (140.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	00000000 (0.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	0000002C (44.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACMODE	=0000000C	78 (1)	
AST	=00000004	79 (1)	
ASTPRM	=00000008	80 (1)	
DS$AX_SOFTPCB	00000000-XR		129 (1)
ENABLE	=00000004	81 (1)	#-135 (1)
EXE$SETAST	00000000-R	128 (1)	
PCB$B_ASTEN	0000000D		133 (1) 135 (1)
PSL$S_CURMOD	=00000C02		#-131 (1)
PSL$V_CURMOD	=00000018		#-131 (1)
SCH$NEWLVL	00000000-XR		136 (1)
SS$_WASCLR	=00000001		#-134 (1)
SS$_WASSET	=00000009		#-132 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	66 (1)	66 (1)
$DEFINI	1	66 (1)	66 (1) 67 (1) 68 (1) 69 (1) 70 (1) 71 (1) 72 (1)
$IPLDEF	1	67 (1)	67 (1)
$PCBDEF	4	68 (1)	68 (1)
$PRDEF	4	69 (1)	69 (1)
$PSLDEF	2	70 (1)	70 (1)
$RSNDEF	1	71 (1)	71 (1)
$SSDEF	21	72 (1)	72 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	39	00:00:00.11	00:00:00.41
Command processing	145	00:00:00.74	00:00:02.38
Pass 1	539	00:00:07.75	00:00:17.88
Symbol table sort	0	00:00:00.98	00:00:01.01
Pass 2	76	00:00:01.18	00:00:01.32
Symbol table output	0	00:00:00.06	00:00:00.07
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	23	00:00:00.10	00:00:00.10
Assembler run totals	841	00:00:10.95	00:00:23.20

The working set limit was 1000 pages.
35934 bytes (71 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 641 non-local and 1 local symbols.
149 source lines were read in Pass 1, producing 0 object records in Pass 2.
40 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	3
DRB1:[DS.WORK]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;955	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	11

786 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DC.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ASTCON/UPDA=(ASTCON.UPD,ASTCON.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 41 HISTORY ; DETAILED
(1) 76.9 Librarys and Equated Symbols
(1) 109.4 Data PSect Declarations
(1) 122 SCH$ASTDEL - AST DELIVERY INTERRUPT HANDLER
(1) 341 SCH$QAST - ENQUEUE AST CONTROL BLOCK FOR PROCESS

```
                                        .TITLE  ASTDEL  *** ASTDEL AST delivery
                                        .IDENT  /07-07/
                                        .NoShow Conditionals                    ;                                       [07]
-3
;
;*****************************************************************************
;*                                                                           *
;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
;*                                                                           *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER     *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
;*  TRANSFERRED.                                                             *
;*                                                                           *
;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
;*  CORPORATION.                                                             *
;*                                                                           *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;*                                                                           *
;*****************************************************************************

;++
; FACILITY: EXECUTIVE, SCHEDULER
;
; ABSTRACT:
;       ASTDEL CONTAINS THE AST DELIVERY INTERRUPT SERVICE ROUTINE AND THE
;       ASSOCIATED SUBROUTINES SCH$QAST AND SCH$NEWLVL.  THESE ROUTINES
;       IMPLEMENT THE PRIMITIVE AST QUEUEING AND DELIVERY MECHANISMS.
;
; ENVIRONMENT:
;       MODE = KERNEL
;--
;
;       .PAGE
;       .SBTTL  HISTORY                         ; DETAILED
;
; AUTHOR:       R. HUSTVEDT     CREATION DATE: 1-SEP-76
;
; MODIFIED BY:
;       V2.04   RIH0082         RICHARD I. HUSTVEDT     04-APR-1980
;               REMOVE ASTPEN FLAG AND REPLACE WITH ACCESS MODE CHECK
;
;       V2.03   LMK0001         LEN KAWELL      23-JAN-1980
;               ADD CHECKS TO AVOID FILLING THE STACK WHEN AST'S ARE
;               CONTINUALLY QUEUED TO A MODE BEFORE IT CAN EXECUTE
;               THE (SECOND) AST DELIVERY REI.
;
;       V2.02   RIH0031         R. I. HUSTVEDT  07-JUL-1979     14:35
;               CHECK FOR AUTOMATIC STACK EXPANSION AND EXPAND USER STACK.
;
;       V2.01   RIH0030         R. I. HUSTVEDT  20-JUL-1979     10:40
;               ADD GLOBAL FOR RETURN ADDRESS FROM AST CALL
```

```
0000     59 ;
0000     60 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
0000     61 ;               Roger Riggs, 27-Jan-1980, Version 5.2
0000     62 ;       01      Fixed so ACB is requeued to head of queue if rejected
0000     63 ;               because ast is already active on this level.
0000     64 ;       02      David Butenhof   06-Feb-80
0000     65 ;               Add routine EXE$SETAST to emulate VMS function of
0000     66 ;               enabling/disabling AST delivery.
0000     67 ;
0000     68 ;               Dave Butenhof   13-may-1980, Version 5.4
0000     69 ;       03      Modify VMS V2.0 source for supervisor environment
0000     70 ;               Dave Butenhof   18-jun-1980, version 5.5
0000     71 ;       04      Remove EXE$SETAST routine from here, to enhance VMS
0000     72 ;               compatibility AST delivery enable/disable will be
0000     73 ;               implemented as VMS does it.
0000     74 ;       05      Dave Butenhof, 23-feb-1981, version 6.3
0000     75 ;               Fix truncation errors
0000     76 ;
0000     .1 ;       06      - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     .2 ;               Changed PSECT SEP to PSECT Data and PSECT Code.
0000     .3 ;               Also added .LIBRARY statements for $DS and $DIAG.
0000     .4 ;
0000     .5 ;       07      Jack Stansbury, 1-June-1982, Version 6.7
0000     .6 ;               Fixed three truncation errors.
0000     .7 ;-
```

```
                                0000     .9          .Subtitle       Librarys and Equated Symbols
-7                              0000     84
                                0000     85 ;
                                0000     86 ; INCLUDE FILES:
                                0000     87 ;
                                0000     88          .LIBRARY        /$DS/           ;                                       [06]
                                0000     89          .LIBRARY        /$DIAG/         ;                                       [06]
                                0000     90                          ;                                                       [06]
-3                              0000     94          CMKDEF                          ; CHMK HOOKS AND SYMBOLS
                                0000     95          $ACBDEF                         ; AST CONTROL BLOCK DEFINITIONS
                                0000     96          $IPLDEF                         ; IPL DEFINITIONS
                                0000     97          $PCBDEF                         ; PCB DEFINITIONS
                                0000     98          $PHDDEF                         ; PHD DEFINITIONS
                                0000     99          $PRDEF                          ; PROCESSOR REGISTER DEFINITIONS
                                C000    100          $PRIDEF                         ; PRIORITY INCREMENT CLASS DEFS
                                0000    101          $PSLDEF                         ; PSL FIELD DEFINITIONS
                                0000    102          $SSDEF                          ; STATUS CODE DEFINITIONS
-3                              0000    106
                                0000    107 ;
                                0000    108 ; EQUATED SYMBOLS:
                                0000    109 ;
                                0000     .1
                     00000000   0000     .2 ASTEXIT=0                               ; AST EXIT CHANGE MODE CODE
```

```
                                0000     .4          .Subtitle       Data PSect Declarations
                           00000000     .5          .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                [06]
                                0000     .6
-1                              0000     111 ;
                                0000     112 ; OWN STORAGE:
                                0000     113 ;
                                0000     114
-2                              0000     117          MODNAM  ASTDEL
                                0007     118
                                0007     119 ASTACKERR:
                     4B 43 41 54 53 00' 0007     120          .ASCIC  .STACK ERROR.
                     52 4F 52 52 45 20  000D
                                08  0007
```

```
-1                                  0013   122          .SBTTL  SCH$ASTDEL - AST DELIVERY INTERRUPT HANDLER
                                 00000000   123          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                [06]
                                     0000   124
                                     0000   125 ;++
                                     0000   126 ; FUNCTIONAL DESCRIPTION:
                                     0000   127 ;       SCH$ASTDEL RECEIVES THE AST DELIVERY INTERRUPT (IPL - 2) WHICH
                                     0000   128 ;       IS INITIATED BY AN REI INSTRUCTION DETECTING ASTLVL LESS THAN
                                     0000   129 ;       OR EQUAL TO PSL<CURRENT_MODE>.  THE HEAD OF THE AST QUEUE
                                     0000   130 ;       FOR THE CURRENT PROCESS IS REMOVED AND PROCESSED.  SPECIAL
                                     0000   131 ;       KERNEL MODE ASTS ARE PROCESSED WITH IPL REMAINING AT IPL 2.
                                     0000   132 ;       NORMAL ASTS ARE DELIVERED BY PUSHING THE AST INFORMATION ON
                                     0000   133 ;       THE STACK OF THE MODE RECEIVING THE AST AND THE AST ACTIVE
                                     0000   134 ;       BIT FOR THAT MODE IS SET TO PREVENT SUBSEQUENT ASTS UNTIL THE
                                     0000   135 ;       CURRENT ONE FOR THAT MODE HAS BEEN PROCESSED.
                                     0000   136 ;       SPURIOUS AST INTERRUPTS WILL BE DETECTED AND IGNORED.
                                     0000   137 ;
                                     0000   138 ;
                                     0000   139 ; CALLING SEQUENCE:
                                     0000   140 ;       IPL - 2  INTERRUPT
                                     0000   141 ;
                                     0000   142 ;
                                     0000   143 ; INPUT PARAMETERS:
                                     0000   144 ;       00(SP) = PC AT AST DELIVERY INTERRUPT
                                     0000   145 ;       04(SP) = PSL AT AST DELIVERY INTERRUPT
                                     0000   146 ;
                                     0000   147 ; IMPLICIT INPUTS:
                                     0000   148 ;       PCB OF CURRENT PROCESS LOCATED VIA SCH$GL_CURPCB
                                     0000   149 ;       AST CONTROL BLOCK AT HEAD OF AST QUEUE FOR PROCESS
                                     0000   150 ;
                                     0000   151 ;
                                     0000   152 ; OUTPUT PARAMETERS:
                                     0000   153 ;       NONE
                                     0000   154 ;
                                     0000   155 ; IMPLICIT OUTPUTS:
                                     0000   156 ;       *** TBS ***
                                     0000   157 ;
                                     0000   158 ; COMPLETION CODES:
                                     0000   159 ;       NONE
                                     0000   160 ;
                                     0000   161 ; SIDE EFFECTS:
                                     0000   162 ;       *** TBS ***
                                     0000   163 ;
                                     0000   164 ;--
                                     0000   165
                                     0000   166          .ALIGN  LONG                            ; INTERRUPT ROUTINES ON LW BOUND
                                     0000   167 SCH$ASTDEL::                                     ; AST DELIVERY INTERRUPT HANDLER
                                     0000   168          PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; SAVE R0-R5
                          00000000'EF DO     0002   169          MOVL    L^SCH$GL_CURPCB, R4     ; Get pointer to current PCB
                                  54         0008
                                     0009   170          SETIPL  #IPL$_SYNCH                     ; BLOCK SYSTEM EVENTS
                               55    10 B4   000C   171 10$:     REMQUE  @PCB$L_ASTQFL(R4),R5    ; AND REMOVE HEAD OF QUEUE
                                     18 1D   0010   172          BVS     ASTDEXIT                ; EXIT IF QUEUE EMPTY
                            4B 08 A5    07 E5   0012   173          BBCC    #ACB$V_KAST,ACB$B_RMOD(R5),NORM ; BR IF NORMAL AST
```

```
                        0017   175 ;
                        0017   176 ;       THE KERNEL AST ROUTINE IS ENTERED VIA A JSB TO THE SPECIFIED
                        0017   177 ;       ADDRESS WITH IPL=2 AND THE POINTER TO THE AST CONTROL BLOCK
                        0017   178 ;       IN R5.  IT IS THE RESPONSIBILITY OF THE KERNEL AST ROUTINE
                        0017   179 ;       TO PROPERLY RELEASE OR OTHERWISE DISPOSE OF THE AST CONTROL
                        0017   180 ;       BLOCK.  THE PCB BASE ADDRESS IS IN R4.
                        0017   181 ;
                        0017   182 ;       REGISTERS R0-R5 HAVE BEEN PRESERVED AND ARE AVAILABLE FOR
                        0017   183 ;       USE BY THE AST ROUTINE.
                        0017   184 ;
                        0017   185         SETIPL  #IPL$_ASTDEL            ; DROP IPL TO PERMIT SYSTEM EVENTS
              18 B5   16 001A   186         JSB     @ACB$L_KAST(R5)         ; DO KERNEL MODE AST
   00000000'EF   D0   001D   187         MOVL    L^SCH$GL_CURPCB, R4     ; Get pointer to current PCB
                 54   0023
   0000002D'EF   16   C024    .1         JSB     SCH$NEWLVL              ; COMPUTE NEW ASTLVL                    [07]
                      002A    .2
-1                    002A   189 ASTDEXIT:                               ; AST DELIVERY EXIT
                3F   BA 002A   190         POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE REGISTERS R0-R5
                     02 002C   191         REI                             ; AND RETURN
```

```
                                002D   193 ;
                                002D   194 ;       SCH$NEWLVL - COMPUTE NEW ASTLVL
                                002D   195 ;
                                002D   196 ; INPUT:
                                002D   197 ;       R4 - ADDRESS OF CURRENT PCB
                                002D   198 ;
                                002D   199 ; OUTPUT:
                                002D   200 ;       PR$_ASTLVL - NEW AST LEVEL
                                002D   201 ;       PHD$V_ASTLVL - NEW AST LEVEL
                                002D   202 ;
                                002D    .1
                                002D   203 SCH$NEWLVL::                                 ; COMPUTE NEW AST LEVEL
     50    10 A4    DE          002D   204         MOVAL   PCB$L_ASTQFL(R4),R0         ; GET ADDRESS OF AST QUEUE
                                0031   205         DSBINT  #IPL$_SYNCH                 ; DISABLE SYSTEM EVENTS
        51    60    D0          C037   206         MOVL    (R0),R1                     ; GET FLINK
        51    50    D1          003A   207         CMPL    R0,R1                       ; AND CHECK FOR EMPTY QUEUE
              19    13          003D   208         BEQL    20$                         ; YES, QUEUE IS EMPTY
        02    00    EF          003F   209         EXTZV   #0,#2,ACB$B_RMOD(R1),R2     ; GET REQUEST MODE
     52    0B A1                0042
  04 0B A1    07    E1          0045   210         BBC     #ACB$V_KAST,ACB$B_RMOD(R1),10$  ; NOT A KERNEL AST
              52    D4          004A   211         CLRL    R2                          ; SET TO KERNEL MODE
              0D    11          004C   212         BRB     30$                         ; FOR ASTLVL
                                004E    .1
  0D A4    0C A4    8B          004E    .2 10$:    BICB3   PCB$B_ASTACT(R4),PCB$B_ASTEN(R4),R0 ; COMPUTE DELIVERABLE
                    50          0053
     03 50    52    E0          0054    .3         BBS     R2,R0,30$                   ; YES, CAN BE DELIVERED
                                0058    .4
        52    04    D0          0058    .5 20$:    MOVL    #4,R2                       ; NONE DELIVERABLE, NULL ASTLVL
                                005B    .6
-3      13    52    DA          005B   216 30$:    MTPR    R2,#PR$_ASTLVL              ; SET ASTLVL REGISTER
                                005E   217         ENBINT                              ; ENABLE SYSTEM EVENTS AGAIN
                    05          0061   218         RSB                                 ; RETURN
```

```
                        0062   220 ;
                        0062   221 ;       AT THIS POINT THE KERNEL STACK IS:
                        0062   222 ;       00(SP) = SAVED R0
                        0062   223 ;       04(SP) = SAVED R1
                        0062   224 ;       08(SP) = SAVED R2
                        0062   225 ;       12(SP) = SAVED R3
                        0062   226 ;       16(SP) = SAVED R4
                        0062   227 ;       20(SP) = SAVED R5
                        0062   228 ;       24(SP) = SAVED PC
                        0062   229 ;       28(SP) = SAVED PSL
                        0062   230 ;
                        0062   231
                        0062   232 NORM:                                        ; NORMAL AST DELIVERY
   02   00   EF        0062    .1          EXTZV   #ACB$V_MODE, -               ; GET AST REQUEST MODE                  [07]
53   0B A5              C065
                        0068    .2                    #ACB$S_MODE, -           ;                                       [07]
                        0068    .3                    ACB$B_RMOD(R5), -        ;                                       [07]
                        0068    .4                    R3                       ;                                       [07]
   02   18   ED        0068    .5          CMPZV   #PSL$V_CURMOD, -             ; IS CURRENT MODE LEGAL                 [07]
53   1C AE              006B
                        006E    .6                    #PSL$S_CURMOD, -         ;                                       [07]
                        006E    .7                    28(SP), -                ;                                       [07]
                        006E    .8                    R3                       ;                                       [07]
        06   18        006E    .9          BGEQ    20$                          ; YES, NOT SPURIOUS
                        0070   .10
10 A4   65   0F        0070   .11 10$:     INSQUE  (R5),PCB$L_ASTQFL(R4)        ; REQUEUE AT HEAD OF QUEUE
        B4   11        0074   .12          BRB     ASTDEXIT                     ; AND EXIT
                        0076   .13
-5      F5 0D A4   53  E1  0076   238 20$:  BBC     R3,PCB$B_ASTEN(R4),10$       ; BR IF AST DISABLED
        F0 0C A4   53  E2  007B   239       BBSS    R3,PCB$B_ASTACT(R4),10$      ; SET AST ACTIVE
        03 0B A5   06  E5  0080   240       BBCC    #ACB$V_QUOTA,ACB$B_RMOD(R5),30$ ; SKIP IF NO QUOTA ACCOUNTING
              38 A4    B6  0085   241       INCW    PCB$W_ASTCNT(R4)             ; UPDATE OUTSTANDING COUNT
                        0088    .1
                        0088    .2 30$:                                         ; AND DELIVER AST
              A2 AF    16  0088    .3       JSB     SCH$NEWLVL                   ; COMPUTE NEW AST LEVEL                 [07]
-2                      008B   244         SETIPL  #IPL$_ASTDEL                 ; NOW DROP IPL TO PERMIT SYSTEM EVENTS
                        008E   245 ;
                        008E   246 ;       A NEW VALUE FOR ASTLVL HAS NOW BEEN COMPUTED AND SET.
                        008E   247 ;       THE AST REPRESENTED BY THE AST CONTROL BLOCK LOCATED VIA
                        008E   248 ;       R5 CAN NOW BE DELIVERED.
                        008E   249 ;
           53   D5      008F   250         TSTL    R3                           ; CHECK FOR DELIVERY TO KERNEL
           40   12      0090   251         BNEQ    NOTKMODE                     ; NOT KERNEL MODE
```

```
                                0092   253 ;
                                0092   254 ;       DELIVER NORMAL AST TO KERNEL MODE
                                0092   255 ;
                     03   BA    0092   256         POPR    #^M<R0,R1>              ; RESTORE R0,R1
                                0094   257 ;
                                0094   258 ;       00(SP) = R2, 04(SP) = R3, 08(SP) = R4, 12(SP) = R.,
                                0094   259 ;       16(SP) = PC, 20(SP) = PSL
                                0094   260 ;
                                0094   261 40$:
                7E   08 AE   7D 0094   262         MOVQ    8(SP),-(SP)             ; SHUFFLE STACK
                                0098   263 ;
                                0098   264 ;       00(SP) = R4, 04(SP) = R5, 08(SP) = R2, 12(SP) = R3,
                                0098   265 ;       16(SP) = R4, 20(SP) = R5, 24(SP) = PC, 28(SP) = PSL
                                0098   266 ;
                7E   08 AE   7D C098   267         MOVQ    8(SP),-(SP)             ; OPEN FOR AST ARG LIST
                                009C   268 ;
                                009C   269 ;       00(SP) = R2, 04(SP) = R3, 08(SP) = R4, 12(SP) = R5,
                                009C   270 ;       16(SP) = R2, 20(SP) = R3, 24(SP) = R4, 28(SP) = R5,
                                009C   271 ;       32(SP) = PC, 36(SP) = PSL
                                009C   272 ;
             18 AE    50   7D 009C   273         MOVQ    R0,24(SP)               ; SET R0,R1 IN ARG LIST
                                00A0    .1
          14 AE    14 A5   D0 00A0   274 50$:    MOVL    ACB$L_ASTPRM(R5),20(SP) ; SET AST PARAMETER IN ARG LIST
             10 AE    05   D0 00A5   275         MOVL    #5,16(SP)               ; SET COUNT FOR ARGUMENT LIST
                50    55   D0 00A9   276         MOVL    R5,R0                   ; RELEASE AST CONTROL BLOCK
                   10 A5   DD 00AC   277         PUSHL   ACB$L_AST(R5)           ; SAVE AST ROUTINE ADDRESS
          00000000'EF   16 00AF    .1           JSB     EXE$DEANONPAGED         ; TO DYNAMIC POOL                       [07]
-1                 3E   BA 00B5   279         POPR    #^M<R1,R2,R3,R4,R5>     ; RESTORE R1-R5
                                00B7   280         SETIPL  #0                      ; DROP IPL TO ZERO
                                00BA   281 ;       BRB     EXE$ASTDEL              ; FALL THROUGH TO CALL AST ROUTINE
```

B 1 Documentation
C 1 Documentation
D 1 Documentation
E 1 Documentation
F 1 Documentation
G 1 Documentation
H 1 Documentation
I 1 Documentation
J 1 Documentation
K 1 Documentation
L 1 Documentation
M 1 Documentation
N 1 Documentation
B 2 Documentation
C 2 Documentation
D 2 Documentation
E 2 Documentation
F 2 Documentation
G 2 Documentation
H 2 Documentation
I 2 Documentation
J 2 Documentation
K 2 Documentation
L 2 Documentation
M 2 Documentation
N 2 Documentation
B 3 Documentation
C 3 Documentation
D 3 Documentation
E 3 Documentation
F 3 Documentation
G 3 Documentation
H 3 Documentation
I 3 Documentation
J 3 Documentation
K 3 Documentation
L 3 Documentation
M 3 Documentation
N 3 Documentation
B 4 Documentation
C 4 Documentation
D 4 Documentation
E 4 Documentation
F 4 Documentation
G 4 Documentation
H 4 Documentation
I 4 Documentation
J 4 Documentation
K 4 Documentation
L 4 Documentation
M 4 Documentation
N 4 Documentation
B 5 Map
C 5 Map
D 5 Map
E 5 Map
F 5 Map
G 5 Map
H 5 Map
I 5 Map
J 5 Map
K 5 Map
L 5 Map
M 5 Map
N 5 Map
B 6 Map
C 6 Map
D 6 Map
E 6 Map
F 6 Map
G 6 Map
H 6 Map
I 6 Map
J 6 Map
K 6 Map
L 6 Map
M 6 Map
N 6 Map
B 7 Map
C 7 Map
D 7 Map
E 7 Map
F 7 Map
G 7 Map
H 7 Map
I 7 Map
J 7 Map
K 7 Map
L 7 Map
M 7 Map
N 7 Map
B 8 Map
C 8 Map
D 8 Map
E 8 Map
F 8 Map
G 8 Map
H 8 Map
I 8 Map
J 8 Map
K 8 Map
L 8 Map
M 8 VAX DIAGNOSTIC
N 8 *** ACPFDT ACP function dispat
B 9 *** ACPFDT ACP function dispat
C 9 *** ACPFDT ACP function dispat
D 9 ACCESS AND CREATE ACP FUNCTION
E 9 DEACCESS ACP FUNCTION PROCESSI
F 9 DELETE AND MODIFY ACP FUNCTION
G 9 MOUNT ACP FUNCTION PROCESSING
H 9 READ AND WRITE BLOCK ACP FUNCT
I 9 READ AND WRITE BLOCK ACP FUNCT
J 9 READ AND WRITE BLOCK ACP FUNCT
K 9 Symbol table
L 9 Symbol table
M 9 Psect synopsis
N 9 Cross reference
B 10 Cross reference
C 10 Cross reference
D 10 *** ANSI magtape RMS support
E 10 *** ANSI magtape RMS support
F 10 *** ANSI magtape RMS support
G 10 ansi$open
H 10 ansi$open
I 10 ansi$open
J 10 ansi$open
K 10 ansi$open
L 10 ansi$open
M 10 ansi$open
N 10 ansi$open
B 11 ansi$open
C 11 ansi$open
D 11 ansi$open
E 11 ansi$open
F 11 ansi$open
G 11 ansi$open
H 11 ansi$open
I 11 ansi$close
J 11 ansi$close
K 11 ansi$cachevbn
L 11 ansi$cachevbn
M 11 ansi$cachevbn
N 11 ansi$cachevbn
B 12 ansi$read
C 12 ansi$read
D 12 ansi$read
E 12 ansi$get
F 12 ansi$get
G 12 ansi$get
H 12 ansi$get
I 12 ansi$get
J 12 ansi$get
K 12 ansi$get
L 12 ansi$get
M 12 *** APT interface to APT contr
N 12 *** APT interface to APT contr
B 13 *** APT interface to APT contr
C 13 DECLARATIONS
D 13 APT SUPPORT ROUTINES
E 13 APT SUPPORT ROUTINES
F 13 APT SUPPORT ROUTINES
G 13 APT SUPPORT ROUTINES
H 13 APT SUPPORT ROUTINES
I 13 APT SUPPORT ROUTINES
J 13 Symbol table
K 13 Symbol table
L 13 Psect synopsis
M 13 Cross reference
N 13 Cross reference
B 14 Cross reference
C 14 Cross reference
D 14 Cross reference
E 14 Cross reference
F 14 *** ASSIGN assign device
G 14 *** ASSIGN assign device
H 14 *** ASSIGN assign device
I 14 ASSIGN I/O CHANNEL
J 14 ASSIGN I/O CHANNEL
K 14 ASSIGN I/O CHANNEL
L 14 ASSIGN I/O CHANNEL

M 14 TEST IF MAILBOX SPECIFIED
N 14 Symbol table
B 15 Symbol table
C 15 Psect synopsis
D 15 Cross reference
E 15 Cross reference
F 15 Cross reference
G 15 *** ASTCON AST control service
H 15 *** ASTCON AST control service
I 15 DECLARATIONS
J 15 EXE$SETAST - SET AST ENABLES
K 15 Symbol table
L 15 Psect synopsis
M 15 Cross reference
N 15 Cross reference
B 16 Cross reference
C 16 *** ASTDEL AST delivery
D 16 *** ASTDEL AST delivery
E 16 HISTORY ; DETAILED
F 16 Librarys and Equated Symbols
G 16 Data PSect Declarations
H 16 SCH$ASTDEL - AST DELIVERY INTE
I 16 SCH$ASTDEL - AST DELIVERY INTE
J 16 SCH$ASTDEL - AST DELIVERY INTE
K 16 SCH$ASTDEL - AST DELIVERY INTE
L 16 SCH$ASTDEL - AST DELIVERY INTE

```
                        00BA   283 ;
                        00BA   284 ;       CALL AST ROUTINE WITH AST ARGUMENT LIST
                        00BA   285 ;
                        00BA   286 ;       THE CALL IS EXECUTED AT THE MODE WHICH RECEIVED THE AST WITH
                        00BA   287 ;       THE AST ARGUMENT LIST ON THE TOP OF THE STACK.  WHEN THE
                        00BA   288 ;       AST ROUTINE RETURNS FROM THE CALL, AN ASTEXIT CHANGE MODE
                        00BA   289 ;       TO KERNEL INSTRUCTION WILL BE ISSUED.  ASTEXIT WILL RESET
                        00BA   290 ;       THE AST ACTIVE BIT FOR THE CURRENT MODE AND MAY CAUSE DELIVERY
                        00BA   291 ;       OF ADDITIONAL ASTS.
                        00BA   292 ;
                        00BA   293 ;       AST ARGUMENT LIST:
                        00BA   294 ;       ------------------
                        00BA   295 ;
                        00BA   296 ;       00(SP) = NUMBER OF ARGUMENTS, =5
                        00BA   297 ;       04(SP) = AST PARAMETER
                        00BA   298 ;       08(SP) = SAVED R0
                        00BA   299 ;       12(SP) = SAVED R1
                        00BA   300 ;       16(SP) = SAVED PC
                        00BA   301 ;       20(SP) = SAVED PSL
                        00BA   302 ;
                        00BA   303 EXE$ASTDEL:                          ; DELIVER AST CALL
61    6E    FA  00BA    304             CALLG   (SP),(R1)               ; CALL AST ROUTINE
5E    08    C0  00BD    305             ADDL    #8,SP                   ; REMOVE ARG COUNT AND ASTPRM
                00C0    306             CMK     ASTEXIT                 ; AND EXIT FROM AST ROUTINE
      03    BA  00CF    307             POPR    #^M<R0,R1>              ; RESTORE R0,R1
            02  00D1    308             REI                             ; EXECUTE REI IN MODE OF AST
```

```
                                00D2   310 ;
                                00D2   311 ;       DELIVER NORMAL AST FOR EXEC, SUPER AND USER MODE
                                00D2   312 ;
                                00D2   313
                                00D2   314 NOTKMODE:                                    ; NOT AN AST FOR KERNEL MODE
                                00D2   315 ;
            51    53    DB      00D2   316          MFPR    R3,R1                       ; GET STACK POINTER
                                00D5   317          IFNOWRT #24,-24(R1),STACKERR,R3     ; ENOUGH STACK SPACE??
         71    18 AE    7D      00DC   318          MOVQ    24(SP),-(R1)                ; MOVE PC,PSL TO PROPER STACK
            71    8E    7D      00E0   319          MOVQ    (SP)+,-(R1)                 ; AND R0,R1 FROM KERNEL STACK
                                00E3    .1
            71    14 A5    D0   00E3   320 10$:     MOVL    ACB$L_ASTPRM(R5),-(R1)      ; SET AST PARAMETER IN ARG LIST
               71    05    D0   00E7   321          MOVL    #5,-(R1)                    ; AND FINALLY, ARGUMENT COUNT OF 5
               53    51    DA   00EA   322          MTPR    R1,R3                       ; SAVE UPDATED STACK POINTER
                  10 A5    DD   00ED   323          PUSHL   ACB$L_AST(R5)               ; STACK AST ENTRY POINT
      14 AE    C7 AF    DE   00F0   324          MOVAL   EXE$ASTDEL,20(SP)           ; SET PC TO AST DELIVERY CALL
               50    55    D0   00F5   325          MOVL    R5,R0                       ; SET ADDRESS OF ACB FOR RELEASE
               53    05    C4   00F8    .1          MULL    #1+<1@<PSL$V_CURMOD-PSL$V_PRVMOD>>, -; CURRENT MODE = PREVIOUS MODE
                                00FB    .2            R3                                ;                                          [07]
               53    16    78   00FB    .3          ASHL    #PSL$V_PRVMOD,R3,24(SP)     ; SYNTHESIZE PSL FOR PROPER MODE
                  18 AE          00FE
         00000000'EF    16      0100    .4          JSB     EXE$DEANONPAGED             ; RELEASE AST CONTROL BLOCK                [07]
-3          3E    BA   0106   329          POPR    #^M<R1,R2,R3,R4,R5>         ; RESTORE R1-R5
                  02    0108   330          REI                                 ; AND ENTER AST MODE
                                0109   331                                      ; DROPS IPL TO ZERO
                                0109   332
                                0109   333 ;
                                0109   334 ;       REFLECT STACK ERROR
                                0109   335 ;
                                0109   336 STACKERR:                                   ; ERROR IN STACK MOVE
                                0109   337          ERRSUP_S  ,ASTACKERR
                     00         011E   338          HALT
                  E8    11     011F   339          BRB     STACKERR
```

```
                     0121   341          .SBTTL  SCH$QAST - ENQUEUE AST CONTROL BLOCK FOR PROCESS
                     0121   342
                     0121   343 ;++
                     0121   344 ;
                     0121   345 ; FUNCTIONAL DESCRIPTION:
                     0121   346 ;       SCH$QAST INSERTS THE AST CONTROL BLOCK SUPPLIED IN THE PROPER
                     0121   347 ;       POSITION BY ACCESS MODE IN THE AST QUEUE OF THE PROCESS SPECIFIED
                     0121   348 ;       BYT THE PID FIELD OF THE AST CONTROL BLOCK.  AN AST ARRIVAL EVENT
                     0121   349 ;       IS THEN REPORTED FOR THE PROCESS TO REACTIVATE FROM A WAIT STATE
                     0121   350 ;       IF APPROPRIATE.  THE AST CONTROL BLOCK WILL BE RELEASED IMMEDIATELY
                     0121   351 ;       IF THE PID SPECIFIES A NON-EXISTENT PROCESS.
                     0121   352 ;
                     0121   353 ; CALLING SEQUENCE:
                     0121   354 ;       BSB/JSB SCH$QAST
                     C121   355 ;
                     0121   356 ; INPUT PARAMETERS:
                     0121   357 ;       R2 - PRIORITY INCREMENT CLASS
                     0121   358 ;       R5 - POINTER TO AST CONTROL BLOCK
                     0121   359 ;
                     0121   360 ; IMPLICIT INPUTS:
                     0121   361 ;       PCB OF PROCESS IDENTIFIED BY PID FIELD
                     0121   362 ;
                     0121   363 ; OUTPUT PARAMETERS:
                     0121   364 ;       R0 - COMPLETION STATUS CODE
                     0121   365 ;       R4 - PCB ADDRESS OF PROCESS FOR WHICH AST WAS QUEUED
                     0121   366 ;
                     0121   367 ; IMPLICIT OUTPUTS:
                     0121   368 ;       *** TBS ***
                     0121   369 ;
                     0121   370 ; SIDE EFFECTS:
                     0121   371 ;       THE PROCESS IDENTIFIED BY THE PID IN THE AST CONTROL BLOCK
                     0121   372 ;       WILL BE MADE EXECUTABLE IF NOT SUSPENDED.
                     0121   373 ;
                     0121   374 ; COMPLETION CODES:
                     0121   375 ;       SS$_NORMAL   -  NORMAL SUCCESSFUL COMPLETION STATUS
                     0121   376 ;       SS$_NONEXPR  -  NON-EXISTENT PROCESS
                     0121   377 ;
                     0121   378 ;--
                     0121   379
                     0121   380 SCH$QAST::                                    ; ENQUEUE AST FOR PROCESS
          50   0C A5   3C  0121   381          MOVZWL  ACB$L_PID(R5),R0        ; GET PROCESS INDEX FOR AST TARGET
                     0125   382          DSBINT  #IPL$_SYNCH                 ; DISABLE SYSTEM EVENTS
00000000'FF40   D0  0128   383          MOVL    @L^SCH$GL_PCBVEC[R0],R4 ; Look up PCB address
                       54   0132
60 A4   0C A5   D1  0133   384          CMPL    ACB$L_PID(R5),PCB$L_PID(R4) ; CHECK FOR MATCH IN PID
                       12   13  0138   385          BEQL    20$                     ; OK, PID MATCHES
                     013A    .1
                     013A    .2 10$:                                         ; PROCESS NOT FOUND
          50   55   D0  013A    .3          MOVL    R5,R0                   ; RELEASE AST CONTROL BLOCK
    00000000'EF   16  013D    .4          JSB     EXE$DEANONPAGED         ; IF NO SUCH PROCESS                    [07]
-3                   0143   389
                     0143   390          ENBINT                          ; ENABLE INTERRUPTS
 50   08E8 8F   3C  0146   391          MOVZWL  #SS$_NONEXPR,R0         ; SET ERROR STATUS CODE
                       05  0148   392          RSB                             ; AND RETURN
                     014C   393
                     014C   394 20$:                                         ; PROCESS EXISTS
          02   U0   EF  014C    .1          EXTZV   #ACB$V_MODE, -          ; EXTRACT REQUEST ACCESS MODE           [07]
```

```
                50   0B A5        014F
                                  0152    .2                    #ACB$S_MODE, -          ;                                       [07]
                                  0152    .3                    ACB$B_RMOD(R5), -       ;                                       [07]
                                  0152    .4                    R0                      ;                                       [07]
                51   10 A4   DE   0152    .5          MOVAL     PCB$L_ASTQFL(R4),R1     ; POINT TO QUEUE HEADER
                     53   61   D0   0156    .6          MOVL      (R1),R3                 ; AND COPY HEADER ADDRESS
                                  0159    .7
-3                   53   51   D1   0159    398 30$:    CMPL      R1,R3                   ; CHECK FOR END OF QUEUE
                          17   13   015C    399         BEQL      40$                     ; INSERT IF AT END
             0D 0B A5     07   E0   015E    400         BBS       #ACB$V_KAST,ACB$B_RMOD(R5),37$  ; BR IF SPECIAL KERNEL AST
                          02   00   ED   0163    401         CMPZV     #0,#2,ACB$B_RMOD(R3),R0 ; COMPARE ACCESS MODES
                50   0B A3        0166
                          05   14   0169    402         BGTR      37$                     ; IF GTR AT RIGHT PLACE
                                  016B    .1
                     53   63   D0   C16B    .2 35$:     MOVL      (R3),R3                 ; FLINK ON TO NEXT ACB
                          E9   11   016E    .3          BRB       30$                     ; AND TRY AGAIN
                                  0170    .4
             F6 0B A3     07   E0   0170    .5 37$:     BBS       #ACB$V_KAST,ACB$B_RMOD(R3),35$  ; BR TO QUEUE KAST FIFO
                                  0175    .6
-3                                0175    406 40$:                                    ; INSERT IN AST QUEUE
                04 B3     65   0E   0175    407         INSQUE    (R5),@ACB$L_ASTQBL(R3)  ; INSERT AFTER PREVIOUS
                51   10 A4   D0   0179    408         MOVL      PCB$L_ASTQFL(R4),R1     ; ADDRESS OF AST QUEUE HEAD
                          50   D4   017D    409         CLRL      R0                      ; ASSUME KERNEL AST
                     0B A1   95   017F    410         TSTB      ACB$B_RMOD(R1)          ; KERNAL AST REQUESTED?
                          10   19   0182    411         BLSS      50$                     ; AN INTERNAL AST
                02   00   EF   0184    .1          EXTZV     #ACB$V_MODE, -          ; GET MOST PRIVILEGED AST LEVEL         [07]
                50   0B A1        0187
                                  018A    .2                    #ACB$S_MODE, -          ;                                       [07]
                                  018A    .3                    ACB$B_RMOD(R1), -       ;                                       [07]
                                  018A    .4                    R0                      ;                                       [07]
             0D A4    0C A4   8B   018A    .5          BICB3     PCB$B_ASTACT(R4), -     ; COMPUTE INACTIVE ENABLED              [07]
                               51   018F
                                  0190    .6                    PCB$B_ASTEN(R4), -      ;                                       [07]
                                  0190    .7                    R1                      ;                                       [07]
                03 51    50   E1   019C    .8          BBC       R0,R1,80$               ; SKIP IF ACTIVE OR DISABLED
                                  0194    .9
                     13   50   DA   0194    .10 50$:    MTPR      R0,#PR$_ASTLVL          ; ALSO SET ASTLVL REGISTER
                                  0197    .11
-5                                0197    417 80$:    ENBINT                            ; ENABLE INTERRUPTS
                     50   01   3C   019A    418         MOVZWL    #SS$_NORMAL,R0          ; SET SUCCESS STATUS CODE
                               05   019D    419         RSB                               ; AND RETURN
                                  019E    420
                                  019E    421         .END
```

```
$ER                             =  00000002     D
$MODULE                            00000000 R   D   02
ACB$B_RMOD                         0000000B     D
ACB$B_TYPE                         0000000A     D
ACB$C_LENGTH                       00000018     D
ACB$K_LENGTH                       00000018     D
ACB$L_AST                          00000010     D
ACB$L_ASTPRM                       00000014     D
ACB$L_ASTQBL                       00000004     D
ACB$L_ASTQFL                       00000000     D
ACB$L_KAST                         00000018     D
ACB$L_PID                          0000000C     D
ACB$S_MODE                      =  00000002     D
ACB$V_KAST                      =  00000007     D
ACB$V_MODE                      =  0000C000     D
ACB$V_QUOTA                     =  00000006     D
ACB$W_SIZE                         00000008     D
ASTACKERR                          00000007 R   D   02
ASTDEXIT                           0000002A R   D   03
ASTEXIT                         =  00000000     D
CMK$ASTEXIT                        ********   X   03
CMK$_                           =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
DS_ERRSUP                          ********   X   03
EXE$ASTDEL                         000000BA R   D   03
EXE$DEANONPAGED                    ********   X   03
IPL$_ASTDEL                     =  00000002     D
IPL$_SYNCH                      =  00000007     D
NORM                               00000062 R   D   03
NOTKMODE                           000000D2 R   D   03
PCB$B_ASTACT                       0000000C     D
PCB$B_ASTEN                        0000000D     D
PCB$B_PRI                          0000000B     D
PCB$B_PRIB                         0000002F     D
PCB$B_TYPE                         0000000A     D
PCB$B_WEFC                         0000002E     D
PCB$C_LENGTH                       0000008C     D
PCB$K_LENGTH                       0000008C     D
PCB$L_ARB                          00000084     D
PCB$L_ASTQBL                       00000014     D
PCB$L_ASTQFL                       00000010     D
PCB$L_EFC2P                        00000058     D
PCB$L_EFC3P                        0000005C     D
PCB$L_EFCS                         00000050     D
PCB$L_EFCU                         00000054     D
PCB$L_EFWM                         0000004C     D
PCB$L_JIB                          00000078     D
PCB$L_OWNER                        0000001C     D
PCB$L_PHD                          00000064     D
PCB$L_PHYPCB                       00000018     D
PCB$L_PID                          00000060     D
PCB$L_PQB                          0000004C     D
PCB$L_SQBL                         00000004     D
PCB$L_SQFL                         00000000     D
PCB$L_STS                          00000024     D
PCB$L_UIC                          00000088     D
PCB$L_WSSWP                        00000020     D
PCB$L_WTIME                        00000028     D
PCB$Q_PRIV                         0000007C     D
PCB$T_LNAME                        00000068     D
PCB$T_TERMINAL                     00000044     D
PCB$W_APTCNT                       00000030     D
PCB$W_ASTCNT                       00000038     D
PCB$W_BIOCNT                       0000003A     D
PCB$W_BIOLM                        0000003C     D
PCB$W_DIOCNT                       0000003E     D
PCB$W_DIOLM                        00000040     D
PCB$W_GPGCNT                       00000034     D
PCB$W_GRP                          0000008A     D
PCB$W_MEM                          00000088     D
PCB$W_MTXCNT                       0000000E     D
PCB$W_PPGCNT                       00000036     D
PCB$W_PRCCNT                       00000042     D
PCB$W_SIZE                         00000008     D
PCB$W_STATE                        0000002C     D
PCB$W_TMBU                         00000032     D
PR$_ASTLVL                      =  00000013     D
PR$_IPL                         =  00000012     D
PSL$S_CURMOD                    =  00000002     D
PSL$V_CURMOD                    =  00000018     D
PSL$V_IS                        =  0000001A     D
PSL$V_PRVMOD                    =  00000016     D
SCH$ASTDEL                         00000000 RG  D   03
SCH$GL_CURPCB                      ********   X   03
SCH$GL_PCBVEC                      ********   X   03
SCH$NEWLVL                         0000002D RG  D   03
SCH$QAST                           00000121 RG  D   03
SIZ...                          =  00000001     D
SS$_NONEXPR                     =  000008E8     D
SS$_NORMAL                      =  00000001     D
STACKERR                           00000109 R   D   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (140.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000013 (19.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	0000019E (414.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	337 (1)	#-337 (1)
$MODULE	00000000-R	117 (1)	337 (1)
ACB$B_RMOD	0000000B		173 (1) 209 (1) 210 (1) 232.3 (1)
			240 (1) 394.3 (1) 400 (1) 401 (1)
			402.5 (1) #-410 (1) 411.3 (1)
ACB$L_AST	00000010		#-277 (1) #-323 (1)
ACB$L_ASTPRM	00000014		#-274 (1) #-320 (1)
ACB$L_ASTQ3L	00000C04		407 (1)
ACB$L_KAST	00000018		186 (1)
ACB$L_PID	0000000C		#-381 (1) #-384 (1)
ACB$S_MODE	=00000002		#-232.2 (1) #-394.2 (1) #-411.2 (1)
ACB$V_KAST	=00000007		#-173 (1) #-210 (1) #-400 (1) #-402.5 (1)
ACB$V_MODE	=00000030		#-232.1 (1) #-394.1 (1) #-411.1 (1)
ACB$V_QUOTA	=00000006		#-240 (1)
ASTACRERR	00000007-R	119 (1)	337 (1)
ASTDEXIT	0000002A-R	189 (1)	#-172 (1) #-232.12 (1)
ASTEXIT	=00000000	109.2 (1)	
BIT...	=00000018	94 (1)	94 (1)
CMK$ASTEXIT	00000000-XR		#-306 (1)
CMK$_	=00000004	94 (1)	
CMK$_ASTEXIT	=00000000	94 (1)	#-306 (1)
CMK$_CANTIM	=0000000B	94 (1)	
CMK$_HIBER	=00000007	94 (1)	
CMK$_INITSCB	=00000008	94 (1)	
CMK$_LAST	=0000000E	94 (1)	
CMK$_MMENABLE	=00000003	94 (1)	
CMK$_RCONSOLE	=00000001	94 (1)	
CMK$_REFRESH	=00000005	94 (1)	
CMK$_SCHDWK	=00000006	94 (1)	
CMK$_SETIMR	=0000000C	94 (1)	
CMK$_SETIPL	=00000009	94 (1)	
CMK$_SETPRT	=0000000D	94 (1)	
CMK$_SGIPR	=0000000A	94 (1)	
CMK$_TCONSOLE	=00000002	94 (1)	
DS_ERRSUP	00000000-XR		337 (1)
EXE$ASTDEL	000000BA-R	303 (1)	324 (1)
EXE$DEANONPAGED	00000000-XR		277.1 (1) 325.4 (1) 385.4 (1)
IPL$_ASTDEL	=00000002		#-185 (1) #-244 (1)
IPL$_SYNCH	=00000007		#-170 (1) #-205 (1) #-382 (1)
NORM	00000062-R	232 (1)	#-173 (1)
NOTKMODE	000000D2-R	314 (1)	#-251 (1)
PCB$B_ASTACT	0000000C		#-212.2 (1) 239 (1) #-411.5 (1)
PCB$B_ASTEN	0000000D		#-212.2 (1) 238 (1) #-411.6 (1)
PCB$L_ASTQFL	00000010		171 (1) 204 (1) 232.11 (1) 394.5 (1)
			#-408 (1)
PCB$L_PID	00000060		#-384 (1)
PCB$W_ASTCNT	00000038		#-241 (1)
PR$_ASTLVL	=00000013		#-216 (1) #-411.10 (1)
PR$_IPL	=00000012		#-170 (1) #-185 (1) #-205 (1) #-217 (1)
			#-244 (1) #-280 (1) #-382 (1) #-390 (1)

				#-417	(1)		
PSL$S_CURMOD	=00000002			#-232.6	(1)		
PSL$V_CURMOD	=00000018			#-232.5	(1)	#-325.1	(1)
PSL$V_IS	=0000001A	94	(1)	#-306	(1)		
PSL$V_PRVMOD	=00000016			#-325.1	(1)	#-325.3	(1)
SCH$ASTDEL	00000000-R	167	(1)				
SCH$GL_CURPCB	00000000-XR			#-169	(1)	#-187	(1)
SCH$GL_PCBVEC	00000000-XR			#-383	(1)		
SCH$NEWLVL	0000002D-R	203	(1)	187.1	(1)	241.3	(1)
SCH$QAST	00000121-R	380	(1)				
SS$_NONEXPR	=000008E8			#-391	(1)		
SS$_NORMAL	=00000001			#-418	(1)		
STACKERR	00000109-R	336	(1)	#-317	(1)	#-339	(1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	95 (1)	95 (1)
$DEF	1	94 (1)	
$DEFINI	1	95 (1)	100 (1) 101 (1) 102 (1) 95 (1) 96 (1) 97 (1) 98 (1) 99 (1)
$EQU	1	94 (1)	94 (1)
$EQULS1	1	94 (1)	94 (1)
$EQULST	1	94 (1)	94 (1)
$GBLINI	2	94 (1)	94 (1)
$IPLDEF	1	96 (1)	96 (1)
$PCBDEF	4	97 (1)	97 (1)
$PHDDEF	6	98 (1)	98 (1)
$PRDEF	4	99 (1)	99 (1)
$PRIDEF	1	100 (1)	100 (1)
$PSLDEF	2	101 (1)	101 (1)
$PUSHADR	1	337 (1)	337 (1)
$SSDEF	21	102 (1)	102 (1)
$VIELD1	1	94 (1)	
CMK	1	306 (1)	306 (1)
CMKDEF	1	94 (1)	94 (1)
DSBINT	1	205 (1)	205 (1) 382 (1)
ENBINT	1	217 (1)	217 (1) 390 (1) 417 (1)
ERRSUP_S	1	337 (1)	337 (1)
IFNOWRT	1	317 (1)	317 (1)
MODNAM	1	117 (1)	117 (1)
SETIPL	1	170 (1)	170 (1) 185 (1) 244 (1) 280 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.24
Command processing	141	00:00:00.68	00:00:02.79
Pass 1	624	00:00:11.48	00:00:21.82
Symbol table sort	3	00:00:01.29	00:00:02.04
Pass 2	131	00:00:02.76	00:00:04.87
Symbol table output	11	00:00:00.11	00:00:00.45
Psect synopsis output	9	00:00:00.03	00:00:00.03
Cross-reference output	30	00:00:00.33	00:00:00.39
Assembler run totals	987	00:00:16.78	00:00:32.64

The working set limit was 1000 pages.
55740 bytes (109 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 776 non-local and 20 local symbols.
450 source lines were read in Pass 1, producing 0 object records in Pass 2.
62 pages of virtual memory were used to define 25 macros.

```
                                        +----------------------------+
                                        ! Macro library statistics !
                                        +----------------------------+

Macro library name                       Macros defined
------------------                       --------------
DRB1:[DS.WORK]DIAG.MLB;955                             3
DRB1:[DS.WORK]DS.MLB;218                               4
DMA1:[SYSO.SYSMAINT]DS.MLB;218                         0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953                       0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1                          6
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2                      9
TOTALS (all libraries)                                22

990 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ASTDEL/UPDA=(ASTDEL.UPD,ASTDEL.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 136 Libraries, Equated Symbols, Externals
(1) 185 Data Psect Declarations
(1) 253 Work Psect Declarations
(1) 264 DSV$ATTACH Attach command processor
(1) 365 DSX$ATTACH Process ATTACH command
(1) 569 Process invalid generic name error
(1) 578 Process invalid generic name error
(1) 665 Get name processing
(3) 729 Get device attributes
(3) 808 parse device name
(3) 991 PT_INTERPRET Decode PT-descriptor
(3) 1291 DP$EXTRACT Make P-table printable
(4) 1481 GET_LINE Condition getline
(4) 1531 FORMAT_PROMPT Format and load prompt
(4) 1577 DSV$SHOWDEVICE Display device information
(4) 1651 DSV$SHOWSELECT Display information selects
(4) 1703 DSV$DEATTACH Deattach device
(4) 1757 Deattach subtree
(4) 1786 DSV$SELECT Select device for testing
(4) 1959 DSV$DESELECT Deselect device for testing
(5) 2077 SHOW_DEV Routine
(6) 2152 INI$PTABLE Initialize P-table
(6) 2190 Check ambiguous device
(6) 2261 Locate /adapter value
(6) 2321 LOC$PTABLE Routine
(6) 2389 LOC$PTDESC Locate PT-descriptor
(6) 2461 INS$PTABLE Insert P-table

```
0000     1          .Title  ATTACH  *** ATTACH Handle ATTACH command
0000     2          .Ident  /07-27/
0000     3          .Dsabl  GBL
0000     4          .NoShow Conditionals
0000     5
0000     6  ;++
0000     7  ; Copyright (c) 1978, 1982, 1983
0000     8  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     9  ;
0000    10  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    11  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    12  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    13  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    14  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    15  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    16  ; REMAIN IN DEC.
0000    17  ;
0000    18  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    19  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    20  ; CORPORATION.
0000    21  ;
0000    22  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    23  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    24
0000    25  ;++
0000    26  ; Facility:
0000    27  ;
0000    28  ; Abstract:
0000    29  ;
0000    30  ; Environment:
0000    31  ;
0000    32  ; Author: Roger Riggs, Creation date:  10-OCT-1978
0000    33  ;
0000    34  ; Modified by:
0000    35  ;       [02] Roger Riggs, 27-Mar-1980
0000    36  ;               Extended buffer for show devices.
0000    37  ;       [03] Roger Riggs, 10-Apr-1980
0000    38  ;               Added $DS_$ADD and $DS_$COMPLEMENT operations to
0000    39  ;               PT-descriptor interpreter
0000    40  ;       [04] Dave Butenhof, 24-sep-1980, Version 6.1
0000    41  ;               Split DSV$ATTACH routine into DSV$ATTACH control routine
0000    42  ;               and DSX$ATTACH, which is callable by diagnostics and will
0000    43  ;               cause no typeout or console/script reads.  The primary
0000    44  ;               purpose is to support the AUTOSIZER.
0000    45  ;       [05] Dave Butenhof, 19-dec-1980, Version 6.2
0000    46  ;               Implement MCT as NEBULA 'central bus'.  Also generalize
0000    47  ;               'central bus' string compares, and generate error if
0000    48  ;               the wrong 'central bus' name is typed (e.g., 'CMI' for 780).
0000    49  ;       [06] Dave Butenhof, 30-mar-1981, version 6.3
0000    50  ;               Implement new 'hub' structures, cross-implementation-wise:
0000    51  ;               'HUB' and 'PHAD'.
0000    52  ;       [07] Marion Baggett, 2-Oct-1981 Version 6.-
0000    53  ;               Print only device and type fields for /BRIEF switch
0000    54  ;               on SHOW DEVICE, and SHOW SELECT commands.
0000    55  ;       [08] Dave Butenhof, 20-Oct-1981, version 6.-
0000    56  ;               Fix a bug in Ptable descriptor location routine.
0000    57  ;       [09] Dave Butenhof, 22-Oct-1981, version 6.-
```

```
                        0000     58 ;          Add ATTACH co  rol C handler.  Also new Ptable
                        0000     59 ;          descriptor comma ds $DS_$LOGICAL and $DS_$CASE.
                        0000     60 ;      [10] Jack Stansbury, 27-Oct-1981, Version 6.-
                        0000     61 ;          Fixed t uncation errors.
                        0000     62 ;      [11] Dave Butenhof, 06-Nov-1981, Version 6.-
                        0000     63 ;          Implemented SELECT/ADAPTER and DESELECT/ADAPTER.
                        0000     64 ;      [12] Dave Butenhof, 18-Nov-1981, version 6.5
                        0000     65 ;          Implement DEATTACH command.
                        0000     66 ;      [13] Dave Butenhof, 20-Nov-1981, version 6.5
                        0000     67 ;          Convert all PRINT's to use type codes, and delete unused
                        0000     68 ;          message 't_insfmem'.
                        0000     69 ;      [14] Dave Butenhof, 24-Feb-1982, version 6.6
                        0000     70 ;          Convert parsing of Device type field to allow use of
                        0000     71 ;          '_' in the name (e.g., 'CI_NODE').
                        0000     72 ;      [15] Dave Butenhof, 02-Mar-1982, version 6.6
                        0000     73 ;          Correct bug in SELECT/DESELECT: If /Adapter not specified,
                        0000     74 ;          and device is duplicated, only first found is acted on, and
                        0000     75 ;          no message is given.  Due to SELECT sorting, it may not even
                        0000     76 ;          be possible to DESELECT that device (without /Adapter).
                        0000     77 ;          Solution: Give error if no /Adapter on duplicate name.
                        0000     78 ;      [16] Dave Butenhof, 10-Mar-1982, version 6.6
                        0000     79 ;          Correct some truncation errors.
                        0000     80 ;      [17] Dave Butenhof, 07-Apr-1982, version 6.7
                        0000     81 ;          Add support for new Ptable descriptor directive, $DS_$NAME;
                        0000     82 ;          this will allow use of enforced generic naming conventions,
                        0000     83 ;          and inherited device names, while remaining compatible with
                        0000     84 ;          the current schema.  Note that this directive must IMMEDIATELY
                        0000     85 ;          follow the $DS_$INITIALIZE directive, and is treated as an
                        0000     86 ;          extension of same.
                        0000     87 ;      [18] Jack Stansbury, 13-Apr-1982, Version 6.7
                        0000     88 ;          Fixed two truncation errors.
                        0000     89 ;      [19] Dave Butenhof, 06-May-1982, Version 6.8
                        0000     90 ;          Make modifications (very minor) to support splitting
                        0000     91 ;          PTD$M_INHERITED into two separated bits, PTD$M_INHERIT_PRE
                        0000     92 ;          to inherit device prefix, and PTD$M_INHERIT_CON to inherit
                        0000     93 ;          controller designation.
                        0000     94 ;
                        0000     95 ;      [20] Marion Baggett, 6-Oct-1982 Version 6.9
                        0000     96 ;          When inserting PTABLE for RB730, do not re-adjust the RB730
                        0000     97 ;          PTABLE for CSR values.  The RB730 is not on the the unibus.
                        0000     98 ;
                        0000     99 ;      [21] Domenic Andella, 13-Apr-82 Version 6.11
                        0000    100 ;          Added central bus name "ABUS" to CENTRAL_BUS list, and
                        0000    101 ;          changed constant in DSX$ATTACH routine, GET_LINK label
                        0000    102 ;          from 3 to 4 to account for new central bus name.
                        0000    103 ;
                        0000    104 ;      22  John Ciukaj     14-Apr-1983     Version 6.11
                        0000    105 ;          - Inhibit device allocation in user mode during
                        0000    106 ;            Select operation if 'Inhibit Allocate' bit is
                        0000    107 ;            set in CRD flags
                        0000    108 ;
                        0000    109 ;      [23]    Domenic Andella 14-Apr-83       Version 6.11
                        0000    110 ;          Corrected bug in DSX$ATTACH routine, GET_LINK label
                        0000    111 ;          to check all CPU link names.
                        0000    112 ;
                        0000    113 ;      24  M. Baggett      11-MAY-1983     VERSION 6.11
                        0000    114 ;          Changed ptd$m_name to mean alphanumeric.
```

```
0000    115 ;
0000    116 ;       25      Bob Bergazzi    May 16,1983     Version 6.11
0000    117 ;               Changed the order of the .LIB statements.
0000    118 ;
0000    119 ;       26      M. Baggett      10-OCT-1983     VERSION 6.13
0000    120 ;               Change to support HSC50 disk/tape names of the form
0000    121 ;               name$GGannn, where 'name' is a maximum
0000    122 ;               of 6 characters and 'GG' is the generic name and is
0000    123 ;               present with a controller letter and unit number.
0000    124 ;
0000    125 ;       27      Domenic Andella 16-FEB-1984     VERSION 6.13
0000    126 ;               Add code in GET_LINK to add offset of X200 to
0000    127 ;               ptable vector field if attached to SBIA 1 for
0000    128 ;               a xxx CPU.
C000    129 ;
0000    130 ;       [28]    RE MUSE         11 MAY 1984     Version 7.0
0000    131 ;               added code to handle long device names (15 characters)
0000    132 ;               of the form 'node$device'.
0000    133 ;--
0000    134
```

```
                     0000   136          .Subtitle       Libraries, Equated Symbols, Externals
                     0000   137 ;
                     0000   138 ; Libraries
                     0000   139 ;
                     0000   140
                     0000   141 .library        \SYS$LIBRARY:LIB\       ;                                       [25]
                     0000   142 .library        \$DS\                   ;                                       [25]
                     0000   143 .library        \$DIAG\                 ;                                       [25]
                     0000   144
                     0000   145 ;
                     0000   146 ; Equated Symbols:
                     0000   147 ;
                     0000   148 ;        .NLIST
                     0000   149 ;        .NLIST  ME,MEB
                     0000   150 ;        .LIST
                     0000   151          $Ds_PtdDef                      ; Define $DS_$NAME flag codes   [17]
                     0000   152          $Ds_PdDef                       ; Define PD$_ byte codes        [17]
                     0000   153          $DS_HDRDEF
                     0000   154          $DS_HPODEF
                     0000   155          $DS_DSADEF
                     0000   156          $DS_DSDEF
                     0000   157          $PRDEF
                     0000   158          $SSDEF
                     0000   159          $IPLDEF
                     0000   160          CLIDEF
                     0000   161          DSFDEF                  ;                                       [07]
                     0000   162          $ds_dsadef              ; APT mailbox defs                      [09]
                     0000   163          $ds_typedef             ; Define type codes                     [12]
                     0000   164          CRDDef                  ; Define bit definitions for
                     0000   165                                  ; CRD$GL_CRD_Flags                      [22]
                     0000   166
                     0000   167 ;        .NLIST
                     0000   168 ;        .LIST   MEB
                     0000   169 ;        .LIST
                     0000   170
                     0000   171          .EXTRN  DS$GA_SELECTS,  DS$GL_CLIBASE
                     0000   172          .EXTRN  DS$PRINTF,      DS_SUPERLINE,   DS_ERRSUP,      SCRIPT$FLUSH
                     0000   173          .EXTRN  EXE$ALONONPAGED,EXE$DEANONPAGED,DSX$PRINT       ;       [12]
                     0000   174          .EXTRN  SCAN$ALPHANUM,  SCAN$DEVICE,    SCAN$NUMERIC
                     0000   175          .EXTRN  SCAN$SPACES,    SCAN$SYMBOL
                     0000   176          .EXTRN  SYS$FAO,        SYS$ALLOC,      SYS$DALLOC
                     0000   177          .EXTRN  DS$GA_PTABLE,   DS$GA_PTDESC
                     0000   178          .EXTRN  DSR$COMPLETION, DSR$IFLOADDEV
                     0000   179          .EXTRN  DS$GL_FLAGS, KB_CHECK   ;                               [09]
                     0000   180          .extrn  scan$decimal,scan$alpha ;                               [09]
                     0000   181          .extrn  ds$cntrlc, Dsx$Print, Ds$GB_Inhibit_Naming      ;       [17]
                     0000   182          .extrn  DS$GL_CRD_FLAGS         ;                               [22]
00000014             0000   183          HP$C_NAMELEN    == 20           ;                               [28]
```

```
                                        0000   185          .SBTTL  Data Psect Declarations
                                        0000   186
                                    00000000   187          .PSECT  Data, NoExe, Shr, NoWrt, Long          ;                [10]
                                        0000   188          MODNAM  ATTACH
                                        0007   189 T_INVALID:
69 6C 61 76 6E 49 0000000F'010E0000'    0007   190          .ASCID  "Invalid or unknown name.!/!AC? "
6E 77 6F 6E 6B 6E 75 20 72 6F 20 64     0015
3F 43 41 21 2F 21 2E 65 6D 61 6E 20     0021
                                  20    002D
                                        002E   191 T_Bad_Name:                                 ;                [17]
72 72 45 20 3F 3F 00000036'010E0000'    002E   192          .Ascid  \?? Error in !AC!ACname, format should be "!AC"!/!AC? \ ;       [17]
43 41 21 43 41 21 20 6E 69 20 72 6F     003C
74 61 6D 72 6F 66 20 2C 65 6D 61 6E     0048
22 20 65 62 20 64 6C 75 6F 68 73 20     0054
   20 3F 43 41 21 2F 21 22 43 41 21     0060
                                        006B   193 T_Part_Of:                                 ;                [17]
      20 66 6F 20 74 72 61 70 20 00'    006B   194          .Ascic  \ part of \                ;                [17]
                                  09    006B
                                        0075   195 T_UNIT_EXCESS:
6E 20 74 69 6E 55 0000007D'010E0000'    0075   196          .ASCID  "Unit number too big or not allowed.!/!AC? "
69 62 20 6F 6F 74 20 72 65 62 6D 75     0083
6C 6C 61 20 74 6F 6E 20 72 6F 20 67     008F
20 3F 43 41 21 2F 21 2E 64 65 77 6F     009B
                                        00A7   197 T_MAIN_BUS:
20 65 72 65 68 54 000000AF'010E0000'    00A7   198          .ASCID  "There is no !AD on this type processor.!/!AC? "
6E 6F 20 44 41 21 20 6F 6E 20 73 69     00B5
70 20 65 70 79 74 20 73 69 68 74 20     00C1
21 2F 21 2E 72 6F 73 73 65 63 6F 72     00CD
                        20 3F 43 41     00D9
                                        00DD   199 T_PROMPT:
   20 3F 43 41 21 000000E5'010E0000'    00DD   200          .ASCID  "!AC? "
65 6D 61 4E 20 65 63 69 76 65 44 00'    00EA   201 T_NAME: .ASCIC  "Device Name"
                                  0B    00EA
6B 6E 69 4C 20 65 63 69 76 65 44 00'    00F6   202 T_LINK: .ASCIC  "Device Link"
                                  0B    00F6
65 70 79 74 20 65 63 69 76 65 44 00'    0102   203 T_TYPE: .ASCIC  "Device type"
                                  0B    0102
                                        010E   204 T_NO_SUCH_DEV:
76 65 64 20 68 63 75 73 20 6F 4E 00'    010E   205          .ASCIC  "No such device !AD:!/"
      2F 21 3A 44 41 21 20 65 63 69     011A
                                  15    010E
                                        0124   206 t_no_such_adapter:                          ;                [11]
61 64 61 20 68 63 75 73 20 6F 4E 00'    0124   207          .ascic  "No such adapter !AD:!/"       ;                [11]
   2F 21 3A 44 41 21 20 72 65 74 70     0130
                                  16    0124
                                        013B   208 t_no_such_on_adapter:                       ;                [11]
76 65 64 20 68 63 75 73 20 6F 4E 00'    013B   209          .ascic  "No such device !AD: on adapter !AD:!/" ;                [11]
20 6E 6F 20 3A 44 41 21 20 65 63 69     0147
3A 44 41 21 20 72 65 74 70 61 64 61     0153
                              2F 21     015F
                                  25    013B
                                        0161   210 T_Ambig_Adap:                                ; Ambigous adapter name  [15]
74 61 63 69 6C 70 75 44 20 3F 3F 00'    0161   211          .AsciC  "?? Duplicate adapter name is illegal (!AS)!/"  ;        [15]
61 6E 20 72 65 74 70 61 64 61 20 65     016D
61 67 65 6C 6C 69 20 73 69 20 65 6D     0179
         2F 21 29 53 41 21 28 20 6C     0185
                                  2C    0161
```

```
                                                018E   212 T_Ambig_Dev:                                ; Ambiguous device name                [15]
65 73 75 20 74 73 75 4D 20 3F 3F 00'            018E   213         .AscIC  "?? Must use /Adaptcr with duplicated device name !AS!/" ;      [15]
69 77 20 72 65 74 70 61 64 41 2F 20             019A
65 74 61 63 69 6C 70 75 64 20 68 74             01A6
6D 61 6E 20 65 63 69 76 65 64 20 64             01B2
                  2F 21 53 41 21 20 65          01BE
                                    36          018E
                                                01C5   214 t_deattach:                               ;                                     [12]
65 65 62 20 73 61 68 20 53 41 21 00'            01C5   215         .ascic  "!AS has been de-attached.!/"   ;                             [12]
65 68 63 61 74 74 61 2D 65 64 20 6E             01D1
                        2F 21 2E 64             01DD
                                    1B          01C5
                                                01E1   216 T_DEVALLOC:
61 63 6F 6C 6C 41 20 3A 44 41 21 00'            01E1   217         .ASCIC  "!AD: Allocated to another user!/"
68 74 6F 6E 61 20 6F 74 20 64 65 74             C1ED
           2F 21 72 65 73 75 20 72 65           01F9
                                    20          01E1
                                                0202   218 T_SHOWDEV:
21 5F 21 43 41 21 5F 21 53 41 21 00'            0202   219         .ASCIC  "!AS!_!AC!_!AS!_!XL  !AS!/"
53 41 21 20 20 4C 58 21 5F 21 53 41             020E
                              2F 21             021A
                                    19          0202
                                                021C   220 T_SHOWDEVB:                              ;                                     [07]
   2F 21 43 41 21 5F 21 53 41 21 00'            021C   221         .ASCIC  "!AS!_!AC!/"              ;                                     [07]
                                    0A          021C
70 20 72 6F 66 20 65 75 6C 61 56 00'            0227   222 T_PNF:  .ASCIC  'Value for prompt "!AC" not found in script!/'  ;             [09]
20 22 43 41 21 22 20 74 70 6D 6F 72             0233
6E 69 20 64 6E 75 6F 66 20 74 6F 6E             023F
           2F 21 74 70 69 72 63 73 20           024B
                                    2C          0227
68 74 20 67 6E 69 70 70 69 6B 53 00'            0254   223 T_SKIP: .ASCIC  'Skipping this line!/'  ;                                 [09]
           2F 21 65 6E 69 6C 20 73 69           0260
                                    14          0254
                                                0269   224 T_SHOULDBE:
      65 62 20 64 6C 75 6F 68 73 00'            0269   225         .ascic  \should be\               ;                                     [09]
                                    09          0269
                                                0273   226 t_yesno:                                 ;                                     [09]
73 65 59 2C 6F 4E 0000027B'010E0000'            0273   227         .ascid  \No,Yes\                  ;                                     [09]
                                                0281   228 T_STRING:
6E 6F 20 43 41 21 00000289'010E0000'            0281   229         .ASCID  "!AC one of !AS!/!AC? "
41 21 2F 21 53 41 21 20 66 6F 20 65             028F
                        20 3F 43                029B
                                                029E   230 Q_HUB:                                   ;                                     [11]
         42 55 48 000002A6'010E0000'            029E   231         .ASCID  "HUB"                     ; Name of HUB device                  [11]
                                                02A9   232
                                                02A9   233 part_list:                               ; List of generic name parts          [17]
                                    05'         02A9   234         .byte   10$-part_list             ; Offset to 'generic'                 [17]
                                    05'         02AA   235         .byte   10$-part_list             ;   same                              [17]
                                    0F          02AB   236         .byte   20$-part_list             ; Offset to 'controller'              [17]
                                    15'         02AC   237         .byte   30$-part_list             ; Offset to 'unit'                    [17]
                                    1D'         02AD   238         .byte   40$-part_list             ; Offset to ''                        [17]
            63 69 72 65 6E 65 67 00'            02AE   239 10$:     .ascic  \generic\                 ;                                     [17]
                                    07          02AE
   72 65 6C 6C 6F 72 74 6E 6F 63 00'            02B6   240 20$:     .ascic  \controller\              ;                                     [17]
                                    0A          02B6
                     74 69 6E 75 00'            02C1   241 30$:     .ascic  \unit\                    ;                                     [17]
                                    04          02C1
```

```
                        00' 02C6   242 40$:     .ascic  \\                      ;                                                       [17]
                        00  02C6
                            02C7   243
                            02C7   244 CENTRAL_BUS:                                 ; List of 'central bus' names
                        05  02C7   245         .BYTE   5                           ; Number of bus names                           [21]
                 49 42 53 00' 02C8   246         .ASCIC  "SBI"                      ; STAR - CPU type 1
                        03  02C8
                 49 4D 43 00' 02CC   247         .ASCIC  "CMI"                      ; COMET - CPU type 2
                        03  02CC
              44 41 48 50 00' 02D0   248         .ASCIC  "PHAD"                     ; PHysical Address Decoder                      [06]
                        04  02D0
              53 55 42 41 00' 02D5   249         .ASCIC  "ABUS"                     ; xxx - CPU type 4                              [21]
                        04  02D5
                 42 55 48 00' 02DA   250         .ASCIC  "HUB"                      ; system HUB device                             [06]
                        03  02DA
                            02DE   251
```

```
                            02DE   253          .SBTTL  Work Psect Declarations
                            02DE   254
                        00000000   255          .PSECT  Work, Noshr, Noexe, Wrt, Long  ;                                [07]
                    00000000 0000  256          FLG$V_BRIEF     = 0                 ; Flag for brief form of commands   [07]
                            0000   257 COMM_FLAGS:                                 ;                                   [07]
                    00000001 0000  258          .BLKB   1                           ; Storage area for /BRIEF flag      [07]
                            0001   259 abort_attach:                               ;                                   [09]
                    00000002 0001  260          .blkb   1                           ; Use to abort ATTACH cmd on ^C     [09]
                            0002   261 q_adapter::                                 ;                                   [11]
                    0000000A 0002  262          .blkq   1                           ; Descriptor for link device name   [11]
```

```
                     000A   264          .SBTTL  DSV$ATTACH      Attach command processor
                     000A   265
                 00000000   266          .PSECT  Code, Exe, Shr, NoWrt, Long              ;                    [10]
                     0000   267 ;++
                     0000   268 ;
                     0000   269 ; Functional Description:
                     0000   270 ;       This routine is called from CLI to finish processing an
                     0000   271 ;       ATTACH command
                     0000   272 ;
                     0000   273 ; Calling Sequence:
                     0000   274 ;       BSBW    DSV$ATTACH
                     0000   275 ;
                     0000   276 ; Input Parameters:
                     0000   277 ;       NONE
                     0000   278 ;
                     0000   279 ; Implicit Inputs:
                     0000   280 ;       R2      Base of CLI data base, DS$GL_CLIBASE
                     0000   281 ;       CLI$Q_FILE      Length,address of remainder of command line
                     0000   282 ;
                     0000   283 ; Output Parameters:
                     0000   284 ;       NONE
                     0000   285 ;
                     0000   286 ; Implicit Outputs:
                     0000   287 ;       New Device Parameter Block or modifications to previous one
                     0000   288 ;
                     0000   289 ; Side Effects:
                     0000   290 ;       NONE
                     0000   291 ;--
```

```
                                    0000   293
                                    0000   294 DSV$ATTACH::
        00000001'EF    94           0000   295         clrb    L^abort_attach          ; Clear abort flag                          [09]
                                    0006   296         $ds_cntrlc_s    set_abort       ; Set up ^C handler                         [09]
        00000000'EF    16           0015   297         JSB     SCRIPT$FLUSH            ; Flush scripts if this console command [10]
            7E    53   7D           001B   298         MOVQ    R3,-(SP)                ; Save R3 and R4
                  55   DD           001E   299         PUSHL   R5                      ; Save R5
               53    7E   DE        0020   300         MOVAL   -(SP),R3                ; Allocate location and set ptr
     5E 00000064    8F   C2        0023   301         SUBL2   #100,SP                 ; Allocate stack buffer
            54    6E   9E           002A   302         MOVAC   (SP),R4                 ;  and set pointer to it
     5E 0000006C    8F   C2        002D   303         SUBL2   #108,SP                 ; Allocate ATTACH prompt string
  04 AE    08    AE   9E           0034   304         MOVAB   8(SP),4(SP)             ; Set string descriptor address
            55    6E   7E           0039   305         MOVAQ   (SP),R5                 ;  and remember where it is
                  3C   BB           003C   306         PUSHR   #^M<R2,R3,R4,R5>        ; Save regs destroyed by MOVC3
              08  A2   28           003E   307         MOVC3   CLI$Q_FILE(R2),-        ; Copy command line to local
            64    0C   B2           0041   308                 @CLI$Q_FILE+4(R2),(R4) ;  buffer (for later appends)
                  3C   BA           0044   309         POPR    #^M<R2,R3,R4,R5>        ; Restore registers
            0C    A2   64           0046   310         MOVAB   (R4),CLI$Q_FILE+4(R2)   ; Correct buffer pointer
         65   0064    8F   3C       004A   311 10$:     MOVZWL  #100,(R5)               ; Re-set length of buffer
                  65   7F           004F   312         PUSHAQ  (R5)                    ; Address of return prompt
              08  A2   7F           0051   313         PUSHAQ  CLI$Q_FILE(R2)          ; Address of command descriptor
        010A'CF    02   FB           0054   314         CALLS   #2,W^DSX$ATTACH         ; Try to do the attach
              74   50   E8           0059   315         BLBS    R0,40$                  ; Exit loop if success
0000FE00'EF    1F   E0              005C   316         bbs     #dsa$v_apt, -           ; If APT mode, abort processing             [09]
                  30                0063   317                 L^dsa$gl_flags, 20$     ; ...with error (fatal)                     [09]
                  15   E1           0064   318         BBC     #DS$V_SCRIPT,-          ;                                           [07]
  2B 00000000'EF                    0066   319                 L^DS$GL_FLAGS,30$       ; Branch if not script running              [07]
            04    B5   9F           006C   320         PUSHAB  @4(R5)                  ; Address of prompt ASCIC string            [07]
     00000227'EF    9F              006F   321         PUSHAB  L^T_PNF                 ; Edit string for prompt not found          [07]
                  01   DD           0075   322         pushl   #ds$k_printf            ; Use Printf                                [13]
                  15   DD           0077   323         pushl   #ds$k_type_command_err  ; And command error type code               [13]
     00000000'EF    04   FB        0079   324         CALLS   #4, L^DSX$PRINT         ; Type it                                   [13]
     00000254'EF    9F              0080   325         PUSHAB  L^T_SKIP                ; Edit string for skipping                  [07]
                  01   DD           0086   326         pushl   #ds$k_printf            ; Use Printf                                [13]
                  15   DD           0088   327         pushl   #ds$k_type_command_err  ; Command error also                        [13]
     00000000'EF    03   FB        008A   328         CALLS   #3, L^DSX$PRINT         ; Type it.                                  [13]
               003C    31           0091   329         BRW     40$                     ; exit                                      [07]
               0052    31           0094   330 20$:    brw     50$                     ; Waystation to 50$                         [13]
                                    0097   331 30$:                                   ;                                           [07]
        00000000'EF    16           0097   332         jsb     L^kb_check              ; Check for ^C's                            [09]
     2C 00000001'EF    E8           009D   333         blbs    abort_attach, 40$       ; Return directly if so                     [09]
               04    B5   9F        00A4   334         PUSHAB  @4(R5)                  ; Address of prompt ASCIC string
7E 00000064    8F   08    A2  C3     00A7   335         SUBL3   CLI$Q_FILE(R2),#100,-(SP); Length of remaining buffer
                  63   DF           00B0   336         PUSHAL  (R3)                    ; Where to return length read
            51   08 B244   9E        00B2   337         MOVAB   @CLI$Q_FILE(R2)[R4],R1  ; Point to end of current string
                 81   20   90        00B7   338         MOVB    #^A" ",(R1)+            ; Separate new argument by space
              08  A2   B6           00BA   339         INCW    CLI$Q_FILE(R2)          ; Increase length
                  61   9F           00BD   340         PUSHAB  (R1)                    ; Calculate address to read to
     00000000'EF    04   FB        00BF   341         CALLS   #4,L^DS_SUPERLINE       ; Read a line
               07   50   E9        00C6   342         BLBC    R0,40$                  ; Can't continue if failure
            08  A2   63   C0        00C9   343         ADDL2   (R3),CLI$Q_FILE(R2)     ; Set new input line size
                FF7A    31           00CD   344         BRW     10$                     ; Try again                                 [13]
     5E 000000D4    8F   C0        00D0   345 40$:     ADDL2   #212,SP                 ; Clear off buffers
               55   8E   D0        00D7   346         MOVL    (SP)+,R5                ; Restore registers
               53   8E   7D        00DA   347         MOVQ    (SP)+,R3
                                    00DD   348         $ds_cntrlc_s                    ; Clear out handler                         [09]
                                    00E8   349         RSB                             ; Return to caller
```

```
                                00E9    350
                                00E9    351 50$:                                        ; ATTACH error in APT mode                  [09]
                                00E9    352             errsup_s        msgadr=@4(r5)   ; ERRSUP w/ returned message             [09]
                          D3    11  00FB  353             brb     40$                     ; Exit routine if continued                 [09]
                                00FD    354
                                00FD    355 ;
                                00FD    356 ; Control C handler for DSV$ATTACH.  Intercepts ^C and causes quick return
                                00FD    357 ; to CLI.  Putting a KB_CHECK call in the loop would work as well: but this
                                00FD    358 ; way causes routine cleanup, so CONTINUE will not continue ATTACH cmd.
                                00FD    359 ;
                           0000 00FD    360 .entry  set_abort, ^M<>                     ;                                           [09]
          00000001'EF      01   90  00FF  361             movb    #1, abort_attach        ; Set abort flag                            [09]
                   50      01   D0  0106  362             movl    #1, r0                  ; Return success                            [09]
                                04  0109   363             ret                             ;                                           [09]
```

```
                        .SBTTL  DSX$ATTACH      Process ATTACH command

;++
;
; Functional description:
;       This routine is called from DSV$ATTACH to translate an attach
;       command line using a PTABLE descriptor, and create a PTABLE
;       from this data.  If any argument is missing or invalid, it will
;       return with the given prompt string buffer set to the correct
;       prompt to get more data from the operator, and the input line
;       descriptor re-set to exclude the remaining (if any) incorrect
;       text of the input line.
;
; Calling sequence:
;       DS$ATTACH (cmdline, [prompt])
;
; Input Parameters:
;       cmdline         pointer to command line descriptor
;       prompt          pointer to ASCIC-type buffer (with length of remainder
;                       of buffer in byte 1)
;
; Implicit inputs:
;       none
;
; Output Parameters
;       prompt          the buffer is filled with an ASCIC string suitable for
;                       prompting the operator for the next ATTACH parameter.
; Implicit outputs:
;       none
;
; Side Effects:
;       If successfull, creates a PTABLE in dynamic pool space
;
; Register usage:
;       R4,R5           Descriptor of remaining input string
;       R6              Pointer to prompt string descriptor
;       R8              Pointer to input descriptor
;       R10             PTABLE descriptor "PC" for decode
;       R11             Pointer to PTABLE
;
; Return codes:
;       DS$_BADTYPE     Device type argument is bad (no PTABLE descriptor)
;       DS$_BADLINK     Device given as link is not attached
;       DS$_ILLUNIT     Unit number not given or was illegal (too big)
;       DS$_DEVNAME     Invalid device name given
;       SS$_BADPARAM    Number was in bad radix or out of range
;       SS$_INSFARG     Too few arguments given to complete ATTACH
;--
```

```
                          010A   414
                          010A   415          $OFFSET 4,POSITIVE,<-
                          010A   416                  <ATC$_CMDLINE,4>,-
                          010A   417                  <ATC$_PROMPT,4>>
                          0004       ATC$_CMDLINE:
                          0008       ATC$_PROMPT:
                          010A   418
                          010A   419          $OffSet 0, NEGATIVE, <-          ; Offsets for local data                    [17]
                          010A   420                  <Name_Flags, 1>, -       ; .. Name flags from $DS_$NAME              [17]
                          010A   421                  <Parent_Controller, 1>, - ; .. Controller letter of parent           [17]
                          010A   422                  <Max_Unit, 1>, -         ; .. Maximum unit number allowed            [17]
                          010A   423                  <PtDesc, 4>, -           ; .. Address of Ptable descriptor           [17]
                          010A   424                  <Parent_Ptable, 4>, -    ; .. Address of Parent Ptable               [17]
                          010A   425                  <Generic, 10>, -         ; .. Generic name string from $DS_$NAME     [17]
                          010A   426                  <Local_Block> -          ; Size of local block                       [17]
                          010A   427                  >                        ;                                           [17]
                          FFFF       Name_Flags:
                          FFFE       Parent_Controller:
                          FFFD       Max_Unit:
                          FFF9       PtDesc:
                          FFF5       Parent_Ptable:
                          FFEB       Generic:
                          FFE7       Local_Block:
                          010A   428
                    0FFC  010A   429 .ENTRY  DSX$ATTACH, ^M<R2,R3,R4,R5.R6,R7,R8,R9,R10,R11>
                          010C   430
          5E   E7 AD   9E 010C   431          MovAB   Local_Block(Fp), Sp      ; Allocate local storage block              [17]
          58   04 BC   7E 0110   432          MOVAQ   @ATC$_CMDLINE(AP),R8     ; Point to command line desc.
               02 A8   B4 0114   433          CLRW    2(R8)                    ; Clear length used so far
             54   68   7D 0117   434          MOVQ    (R8),R4                  ; Set up register descriptor
             02   6C   91 011A   435          CMPB    (AP),#2                  ; Both arguments given?
                  06   19 011D   436          BLSS    10$                      ; No--fake prompt buffer
          56   08 BC   7E 011F   437          MOVAQ   @ATC$_PROMPT(AP),R6      ; Set up prompt pointer
                  07   12 0123   438          BNEQ    20$                      ; Branch if specified
                  7E   3F 0125   439 10$:     PUSHAW  -(SP)                    ; Allocate fake buffer
                  02   DD 0127   440          PUSHL   #2                       ; Finish descriptor
             56   6E   DE 0129   441          MOVAL   (SP),R6                  ; Save pointer to descriptor
                       012C   442 20$:
                  5B   D4 012C   443          CLRL    R11                      ; Clear PT address
                  1D   11 012E   444          BRB     GET_TYPE                 ; Get hardware device type
                       0130   445
                       0130   446 BAD_TYPE:
    00000102'EF   9F 0130   447          PUSHAB  L^T_TYPE                 ; Set prompt to return                      [10]
    00000007'EF   7F 0136   448          PUSHAQ  L^T_INVALID              ; Error message                             [10]
00000A75'EF   02 FB 013C   449          CALLS   #2,L^FORMAT_PROMPT       ; Format it for return                      [10]
50  006600E8 8F   D0 0143   450          MOVL    #DS$_BADTYPE,R0          ; Set return code
         02C3   31 014A   451          BRW     ERROR_EXIT               ; Return
                       014D   452
                       014D   453 GET_TYPE:
53  00000102'EF   9E 014D   454          MOVAB   L^T_TYPE,R3              ; Address of prompt string                  [10]
    00000A4D'EF   16 0154   455          Jsb     L^GET_LINE               ; Get something from operator               [18]
            36 50   E9 015A   456          BLBC    R0,10$                   ; Exit if error on input
                       015D   457
    00000000'EF   16 015D   458          Jsb     L^SCAN$SYMBOL            ; Scan the hardware type                    [16]
            57   50 3C 0163   459          MOVZWL  R0,R7                    ; Save length of type string
                  C8   13 0166   460          BEQL    BAD_TYPE                 ; Branch if not there
                       0168   461
```

```
                7E    50    7D  0168   462          MOVQ    R0,-(SP)                ; Save descriptor of type
          0000100A'EF   16    16B  463          Jsb     L^LOC$PTDESC            ; Locate descriptor R0/R1          [18]
                BD    13    0171  464          BEQL    BAD_TYPE                ; Branch if not found
                            0173  465
             02 A8    57    A0  0173  466          ADDW2   R7,2(R8)                ; Add type string to length used
                5A    50    D0  0177  467          MOVL    R0,R10                  ; Save address of hardware type block
             F9 A5    50    D0  017A  468          Movl    R0, PtDesc(Fp)          ; Save it again                    [17]
                50    8A    9A  017E  469          MOVZBL  (R10)+,R0               ; Get length of ASCIC name
                5A    50    C0  0181  470          ADDL    R0,R10                  ; Skip name
                51    8A    9A  0184  471          MOVZBL  (R10)+,R1               ; Fetch size of PT
                51    14    C0  0187  472          ADDL2   #HP$C_NAMELEN,R1        ; make room for long device name   [28]
          00000000'EF   16    18A  473          Jsb     L^EXE$ALONONPAGED       ; Allocate a block for this PT     [18]
                03 50   E8  0190  474          BLBS    R0,20$                  ; Branch if enough memory
                            0193  475 10$:
                027A    31  C193  476          BRW     ERROR_EXIT
                            0196  477 20$:
                5B    52    D0  0196  478          MOVL    R2,R11                  ; Copy address of block
          50    51    FD 8F   78  0199  479          ASHL    #-3,R1,R0               ; Number of quadwords present
                82    7C  019E  480 30$:     CLRQ    (R2)+                   ; Clear one
                   FB 50    F5  01A0  481          SOBGTR  R0,30$                  ; Branch if not done yet
             08 AB    51    B0  01A3  482          MOVW    R1,HP$W_SIZE(R11)       ; Save true length of block
                            01A7  483
             52    26 AB   9E  01A7  484          MOVAB   HP$T_TYPE(R11),R2       ; Point to type storage location
                82    6E    90  01AB  485          MOVB    (SP),(R2)+              ; Store length
             57    04 AE   D0  01AE  486          MOVL    4(SP),R7                ; Set address to copy from
                50    6E    3C  01B2  487          MOVZWL  (SP),R0                 ; Set length to copy
                      8E    7C  01B5  488          CLRQ    (SP)+                   ; Remove descriptor
                            01B7  489
                82    87    90  01B7  490 40$:     MOVB    (R7)+,(R2)+             ; Copy byte
                   FA 50    F5  01BA  491          SOBGTR  R0,40$                  ; Copy whole string
                      1D    11  01BD  492          BRB     GET_LINK                ; Get device linking this to the processor
                            01BF  493 BAD_LINK:
          000000F6'EF   9F    01BF  494          PUSHAB  L^T_LINK                ; Set prompt to return             [10]
          00000007'EF   7F    01C5  495          PUSHAQ  L^T_INVALID             ; Format string                    [10]
    00000A75'EF   02    FB  01CB  496          CALLS   #2,L^FORMAT_PROMPT      ; Format prompt for output         [10]
50  006600F0 8F   D0  01D2  497          MOVL    #DS$_BADLINK,R0         ; Set return code
                0234    31  01D9  498          BRW     ERROR_EXIT
                            01DC  499
                            01DC  500 GET_LINK:
53  000000F6'EF   9E    01DC  501          MOVAB   L^T_LINK,R3             ; Prompt line                      [10]
          00000A4D'EF   16    1E3  502          Jsb     L^GET_LINE              ; Get some input                   [18]
                03 50   E8  01E9  503          BLBS    R0,10$                  ; Branch if it worked
                 0221   31  01EC  504          BRW     ERROR_EXIT              ; Take error exit
                            01EF  505 10$:
          00000000'EF   16    1EF  506          Jsb     L^SCAN$DEVICE           ; Get device name of connection    [16]
                57    50    3C  01F5  507          MOVZWL  R0,R7                   ; Save length of link name
                C5    13    01F8  508          BEQL    BAD_LINK                ; Bad device
                            01FA  509
                            01FA  510 ;
                            01FA  511 ; See if the link name typed is one of the pre-uefined 'central bus' names.
                            01FA  512 ; If so, make sure it is the correct one (e.g., 'SBI' for STAR, 'CMI' for
                            01FA  513 ; COMET).  If it is not the correct one, return an error.
                            01FA  514 ;
                            01FA  515
                50    50    3C  01FA  516          MOVZWL  R0,R0                   ; isolate device name length
                007C 8F   BB  01FD  517          PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save registers
                      55    D4  0201  518          CLRL    R5                      ; name index
```

```
                53   000002C7'EF   9E   0203   519          MOVAB   L^CENTRAL_BUS, R3        ; Point to table of names                [10]
                     54     83     9A   020A   520          MOVZBL  (R3)+, R4               ; Get number of names in table
                     52     83     9A   020D   521 20$:     MOVZBL  (R3)+, R2               ; Get length of next name
                            55     D6   0210   522          INCL    R5                      ; Increment type code tried
                     56     51     D0   0212   523          MOVL    R1, R6                  ; Copy input string address
                     50     52     B1   0215   524          CMPW    R2, R0                  ; Same length as input?
                            05     13   0218   525          BEQL    40$                     ; If yes, compare strings
                     53     52     C0   021A   526 30$:     ADDL2   R2, R3                  ; Skip rest of string
                            4A     11   021D   527          BRB     60$                     ; Try next string
                     86     63     91   021F   528 40$:     CMPB    (R3), (R6)+             ; Compare bytes of strings
                            F6     12   0222   529          BNEQ    30$                     ; If no match, try next string
                            53     D6   0224   530          INCL    R3                      ; Advance pointer
                       F6   52     F5   0226   531          SOBGTR  R2, 40$                 ; Loop through string
                     04     55     91   0229   532          CMPB    R5, #4                  ; Is is implementation-specific?         [21]
                            2D     14   022C   533          BGTR    50$                     ; No, accept it unconditionally          [23]
          0000FE17'EF       55     91   022E   534          CMPB    R5, L^DIA$GL_SID+3      ; Match--is it correct one?
                            24     13   0235   535          BEQL    50$                     ; Yes, continue
                   007C     8F     BA   0237   536          POPR    #^M<R2,R3,R4,R5,R6>     ; Restore registers
              000000F6'EF           9F   023B   537          PUSHAB  L^T_LINK                ; Prompt string to return                [10]
                     7E     50     7D   0241   538          MOVQ    R0, -(SP)               ; Descriptor of input name               [10]
              000000A7'EF           7F   0244   539          PUSHAQ  L^T_MAIN_BUS            ; 'There is no !AD on this type pro.'    [10]
          00000A75'EF       04     FB   024A   540          CALLS   #4, L^FORMAT_PROMPT     ; Format it for return                   [10]
          50   006600F0     8F     D0   0251   541          MOVL    #DS$_BADLINK, R0        ; Set return code
                     01B5           31   0258   542          BRW     ERROR_EXIT              ; Error!!!
                   007C     8F     BA   025B   543 50$:     POPR    #^M<R2,R3,R4,R5,R6>     ; Good match.  Restore regs.
                02   A8     57     A0   025F   544          ADDW2   R7, 2(R8)               ; Increase string length used
                  F5   AD     D4   0263   545          ClrL    Parent_Ptable(Fp)       ; Device is connected to HUB             [17]
                     0113           31   0266   546          BRW     GET_NAME                ; Get device name generic                [17]
                   A1     54     F5   0269   547 60$:     SOBGTR  R4, 20$                 ; Try next name string
                   007C     8F     BA   026C   548          POPR    #^M<R2,R3,R4,R5,R6>     ; Restore registers
                                       0270   549
                     52     01     CE   0270   550          mnegl   #1, r2                  ; Set wildcard link for loc$ptable       [09]
              00000FBB'EF           16   0273   551          Jsb     L^LOC$PTABLE            ; Locate P-table for link                [18]
                F5   AD     51     D0   0279   552          MovL    R1, Parent_Ptable(Fp)   ; Save address of parent                [17]
                            03     12   027D   553          BNEQ    70$                     ; Found it
                       FF3D         31   027F   554          BRW     BAD_LINK                ; Can't find link P-table
                02   A8     57     A0   0282   555 70$:     ADDW2   R7,2(R8)                ; Increase length used so far
                2'   AB     51     D0   0286   556          MOVL    R1,HP$A_LINK(R11)       ; Store link address
             18   AB     1C   A1     D0   028A   557          MOVL    HP$A_DVA(R1),HP$A_DEVICE(R11) ; Initialize device address
                                       028F   558
                                       028F   559 ; The following is necessary for xxx to check if the parent link is
                                       028F   560 ; for SBIA 1. If so, an offset of ^200 must be added to the device vectors
                                       028F   561 ; address (SCB second page).
                                       028F   562
          0C   A1     3149535F   8F     D1   028F   563          CMPL    #^A"_SI1",HP$T_DEVICE(R1); Is parent LINK SBIA1?         [27]
                            06     12   0297   564          BNEQ    75$                     ; Branch if not SBIA1                    [27]
             24   AB     0200   8F     A0   0299   565          ADDW    #^X200,HP$W_VECTOR(R11) ; Add offset for SBIA1 VECTORS   [27]
                                       029F   566
                     00DA           31   029F   567 75$:     BrW     GET_NAME                ; Get new device name                    [27]
```

```
                          02A2   569          .Subtitle       Process invalid generic name error
                          02A2   570
                          02A2   571 BAD_UNIT_EXC:
        000000EA'EF    9F 02A2   572          PUSHAB  L^T_NAME                    ; Set prompt text                [17]
        00000075'EF    7F 02A8   573          pushaq  t_unit_excess               ; Unit number too large          [17]
  00000A75'EF    02    FB 02AE   574          CALLS   #2,L^FORMAT_PROMPT          ; Format and store               [17]
50  00660100 8F    D0 02B5   575          movl    #ds$_illunit, r0            ; Illegal unit number            [17]
              0151    31 02BC   576          brw     error_exit                  ; Error                          [17]
```

```
                                02BF    578 .sbttl  Process invalid generic name error
                                02BF    579 ;
                                02BF    580 ; BAD_NAME is branched to if parse_device routine fails.  The Ptable
                                02BF    581 ; descriptor is analysed to print out what the problem was, and what the
                                02BF    582 ; correct format of the name ought to be.  The format it uses is 'GGan'
                                02BF    583 ; where 'GG' is replaced by the Ptable descriptor's generic string (or
                                02BF    584 ; 'gg' if there is none), 'a' represents that the name must have a
                                02BF    585 ; controller letter, and 'n' represents that the name must have a unit
                                02BF    586 ; number.  If the PTD$V_NAME generic flag is set, and PTD$V_CONTROLLER and
                                02BF    587 ; PTD$V_UNIT flags are not, then the name is  an alphanumeric node name of
                                02BF    588 ; 6 or fewer characters.  This is represented  in the error message as
                                02BF    589 ; 'name' instead of the 'GGan' form. If PTD$V_NAME and PTD$V_CONTROLLER
                                02BF    590 ; and/or PTD$V_UNIT flag is set then the generic name is represented as
                                02BF    591 ; name$GGan in the error message.
                                02BF    592 ;
                                02BF    593 ;
                                02BF    594 ;
                                02BF    595 ; Inputs:
                                02BF    596 ;       R2 => return code from parse_device.
                                02BF    597 ;                   0 = no generic found.
                                02BF    598 ;                   1 = bad generic prefix
                                02BF    599 ;                   2 = bad controller (none where should be, or present where
                                02BF    600 ;                       shouldn't be)
                                02BF    601 ;                   3 = bad unit (none where should be, or present where
                                02BF    602 ;                       shouldn't be)
                                02BF    603 ;                   4 = bad node name
                                02BF    604 ;
                                02BF    605 ; This code segment will cause loading of the return prompt string, and error
                                02BF    606 ; exit from DSX$ATTACH.
                                02BF    607 ;
                                02BF    608
                                02BF    609 bad_name:                                   ;                                       [17]
            04    52    D1      02BF    610             cmpl    r2, #4                  ; Validate status                       [17]
                  03    15      02C2    611             bleq    10$                     ; branch if OK                          [17]
            52    04    D0      02C4    612             movl    #4, r2                  ; Knock value down to legal range       [17]
50  000002A9'EF42     90      02C7    613 10$:        movb    part_list[r2], r0       ; Get byte offset of message            [17]
52  000002A9'EF40     9E      02CF    614             movab   part_list[r0], r2       ; Get address of message                [17]
         55    FF AD    9A      02D7    615             movzbl  Name_Flags(Fp), r5      ; Get generic flags                     [17]
               53    6E    9E      02D8    616             movab   (sp), r3               ; Save stack pointer                    [17]
            1B 55    02    E1      02DE    617             bbc     #ptd$v_name, r5, 20$    ; Branch if normal name format          [17]
                                    02E2    618 ;           BITL    #PTD$M_CONTROLLER!PTD$M_UNIT, R5 ; Anything else to check for ? [26]
                                    02E2    619 ;           BNEQ    20$                     ; Branch if more to check.              [26]
  7E    656D616E 8F      D0      02E2    620             movl    #^A"name", -(sp)        ; Push string                           [17]
               7E    04    90      02E9    621             movb    #4, -(sp)               ; ...make it ASCIC                      [17]
  51    000002AD'EF       9A      02EC    622             movzbl  part_list+4, r1         ; Get to the part_list null ASCIC       [17]
51  000002A9'EF41     9E      02F3    623             movab   part_list[r1], r1       ; ...string                             [17]
                  5A    11      02FB    624             brb     90$                     ; Go do formatting, etc.                [17]
                                02FD    625 20$:                                        ; Normal name formatting                [17]
                  54    D4      02FD    626             clrl    r4                      ; Length of string created              [17]
            06 55    00    E1      02FF    627             bbc     #ptd$v_unit, r5, 30$    ; Branch if no unit needed              [17]
            7E    6E 8F    90      0303    628             movb    #^A"n", -(sp)           ; Make string on stack                  [17]
                  54    D6      0307    629             incl    r4                      ; Count the byte                        [17]
               55    01    E1      0309    630 30$:        bbc     #ptd$v_controller, -    ; Branch if                             [17]
                     12            030C    631                      r5, 50$            ; ...no controller needed               [17]
                  5E    D7      030D    632             DECL    SP                      ; Allocate one byte                     [17]
                  54    D6      030F    633             incl    r4                      ; And count it                          [17]
               55    04    E1      0311    634             bbc     #ptd$v_inherit_Con, -   ; Does device inherit parent's name?    [19]
```

```
                    06      0314   635                      r5, 40$                 ; ...if not, use 'a'                    [17]
            6E   FE AD   90  0315   636          MovB        Parent_Controller(Fp), -; Get parent's controller letter        [17]
                         0319   637                      (Sp)                    ; .. onto stack                         [17]
                    04   12  0319   638          bneq        50$                     ; If valid, use it                      [17]
            6E   61 8F   90  031B   639 40$:     movb        #^A"a", (sp)            ; Put controller format on stack        [17]
            51   EB AD   9E  031F   640 50$:     movab       Generic(Fp), r1         ; Get address of generic string         [17]
               50   61   9A  0323   641          MovZBL      (R1), R0                ; Get length of generic string          [26]
                    0B   13  0326   642          beql        70$                     ; Branch if null (use 'gg')             [17]
             7E   6140   90  0328   643 60$:     movb        (r1)[r0], -(sp)         ; Push on bytes in right order          [26]
                    54   D6  032C   644          incl        r4                      ; Count 'em                             [17]
                 F7 50   F5  032E   645          sobgtr      r0, 60$                 ; Do them all                           [17]
                    08   11  0331   646          brb         80$                     ; Finish off                            [17]
          7E   6767 8F   B0  0333   647 70$:     movw        #^A"gg", -(sp)          ; Push on 'gg' string                   [17]
            54   02   C0  0338   648          addl2       #2, r4                  ; And set length                        [17]
       0D 0A AB   02   E1  033B   649 80$:     BBC         #HP$V_NAME,HP$B_FLAGS(R11),85$  ; Check if name needed       [28]
            7E   24   90  0340   650          MOVB        #^A"$",-(SP)            ; Set delimiter                         [26]
    7E   656D616E 8F   D0  0343   651          MOVL        #^A"name", -(SP)        ; Set string                            [26]
            54   05   C0  034A   652          ADDL2       #5, R4                  ; Adjust length                         [26]
            7E   54   90  034D   653 85$:     movb        r4, -(sp)               ; Make string on stack be ASCIC         [17]
51   0000006B'EF   9E  0350   654          movab       t_part_of, r1           ; say 'Error in ? part of name'         [17]
     000000EA'EF   9F  0357   655 90$:     pushab      L^t_name                ; Prompt string                         [17]
              04   AE  9F  035D   656          pushab      4(sp)                   ; Correct name format string            [17]
                   61  9F  0360   657          pushab      (r1)                    ; ' part of ' or ''                     [17]
                   62  9F  0362   658          pushab      (r2)                    ; Failing part of the name              [17]
      0000002E'EF  7F  0364   659          pushaq      t_bad_name              ; Format string for the whole mess      [17]
    0A75'CF   05  FB  036A   660          calls       #5, w^format_prompt     ; Format the output prompt              [17]
50   00660108 8F   D0  036F   661          movl        #dss_devname, r0        ; Set failure reason                    [17]
            5E   53   D0  0376   662          movl        r3, sp                  ; Restore stack                         [17]
                0094   31  0379   663          brw         error_exit              ; return                                [17]
```

```
                                  037C   665          .sbttl  Get name processing
                                  037C   666
                                  037C   667 GET_NAME:
                        00A6   30 037C   668          BsbW    Get_Attributes          ; Set up rest of local block            [17]
 53   000000EA'EF        9E 037F   669          MOVAB   L^T_NAME,R3             ; prompt string                         [10]
      00000A4D'EF        16 0386   670          Jsb     L^GET_LINE              ; Get something                         [19]
               03 50     E8 038C   671          BLBS    R0,10$                  ; Branch if not error
                 007E    31 038F   672          BRW     ERROR_EXIT              ; Take error exit
                         0392   673 10$:
                 0145    30 0392   674          BsbW    Parse_Device            ; Get this new device name              [17]
                   03    12 0395   675          BNeq    20$                     ; Skip if OK                            [17]
                 FF25    31 0397   676          Brw     Bad_Name                ; Branch if invalid                     [17]
                   52    D4 039A   677 20$:     ClrL    R2                      ; Initial unit number                   [17]
    11 FF AD     00    E1 039C   678          BbC     #Ptd$V_Unit, -          ; Branch if no unit allowed             [17]
                         03A1   679                   Name_Flags(Fp), 30$     ;                                       [17]
 52  50   10   10    EF 03A1   680          EXTZV   #16,#16,R0,R2           ; Get unit number                       [26]
       7E   FD AD    9A 03A6   681          MOVZBL  MAX_UNIT(FP),-(SP)      ; Set for longword compare              [26]
         8E   52     D1 03AA   682          CMPL    R2,(SP)+                ; Check full unit number                [26]
                 03    1B 03AD   683          BLeqU   30$                     ; Branch if OK                          [17]
               FEF0    31 03AF   684          BrW     Bad_Unit_Exc            ; Branch to error routine               [17]
    0B AB      52    90 03B2   685 30$:     MovB    R2, Hp$B_Drive(R11)     ; Move to Ptable                        [17]
          50   50    9A 03B6   686          MOVZBL  R0,R0                   ; Isolate length of device name
       02 A8   50    A0 03B9   687          ADDW2   R0,2(R8)                ; Increase length used so far
    0E 0A AB   02    E1 03BD   688          BBC     #HP$V_NAME,HP$B_FLAGS(R11),35$ ; if long device name?          [28]
       52    08 AB    3C 03C2   689          MOVZWL  HP$W_SIZE(R11),R2       ; get size of P-table                   [28]
     52   52   14    C3 03C6   690          SUBL3   #HP$C_NAMELEN,R2,R2     ; determine offset
     52   5B   52    C1 03CA   691          ADDL3   R2,R11,R2               ; address to store long device name     [28]
                 04    11 03CE   692          BRB     37$                     ; continue as before                    [28]
                         03D0   693 35$:
   52    0C AB    9E 03D0   694          MOVAB   HP$T_DEVICE(R11),R2     ; address to store device name in pt
                         03D4   695 37$:
6B   50    01    C1 03D4   696          ADDL3   #1,R0,HP$Q_DEVICE(R11)  ; Store length of ascic name
  04 AB    62    9E 03D8   697          MOVAB   (R2),HP$Q_DEVICE+4(R11) ; Store address of ":" in device name
  82   5F 8F    90 03DC   698          MOVB    #^A":",(R2)+            ; Insert "_" physical device indicator
     82    81    90 03E0   699 40$:     MOVB    (R1)+,(R2)+             ; Copy byte
       FA 50    F5 03E3   700          SOBGTR  R0,40$                  ; Copy whole string
     5A    03    C0 03E6   701          ADDL    #3,R10                  ; Point past max unit number and name
```

```
                                03E9   703
                        7E  54  7D  03E9   704          MOVQ    R4,-(SP)                ; length/address of remaining input
                            66  7F  03EC   705          PUSHAQ  (R6)                    ; Address of prompt descriptor
                          04 AE  7F  03EE   706          PUSHAQ  4(SP)                   ; Address of descriptor
                            5B  DD  03F1   707          PUSHL   R11                     ; Base of P-table
                            5A  DD  03F3   708          PUSHL   R10                     ; Start of PT-desc
                     0612'CF    04  FB  03F5   709          CALLS   #4,W^PT$INTERPRET       ; Do PT-descriptor interpretation
                   02 A8  02 AE  A0  03FA   710          ADDW2   2(SP),2(R8)             ; Include length used
```

```
                5E    08    C0  03FF    712          ADDL    #8,SP                   ; Remove descriptor
                  0B  50    E9  0402    713          BLBC    R0,ERROR_EXIT           ; Take error exit if failed
00001064'EF       6B        FA  0405    714          CALLG   (R11),L^INS$PTABLE      ; Insert it                          [10]
                  01  50    E9  040C    715          BLBC    R0,ERROR_EXIT           ; Failed take error exit
                              04  040F    716          RET
                                  0410    717
                                  0410    718 ERROR_EXIT:
                      50        DD  0410    719          PUSHL   R0                      ; Save error code
```

```
                68   02 A8    3C  0412   721          MOVZWL  2(R8),(R8)              ; Return length used
                   50   5B    D0  0416   722          MOVL    R11,R0                 ; Copy address of buffer
                        06    13  0419   723          BEQL    10$                    ; Skip if none
            00000000'EF       16  0418   724          Jsb     L^EXE$DEANONPAGED      ; Release it                    [18]
                              0421   725 10$:
                        50 8ED0  0421   726          POPL    R0                     ; Restore error code
                              04  0424   727          RET
```

```
                                0425   729          .Subtitle       Get device attributes
                                0425   730
                                0425   731 ;++
                                0425   732 ; Functional Description:
                                0425   733 ;
                                0425   734 ;       This routine is called by DSX$ATTACH to set up the local
                                0425   735 ;       storage block with the information on generic name to
                                0425   736 ;       enforce, maximum number of units, etc.  If running from
                                0425   737 ;       a script, APT, or if there is no $DS_$NAME directive in
                                0425   738 ;       the PTABLE descriptor, it will cause behavior similar to
                                0425   739 ;       the previous ATTACH--i.e., there will be no enforced
                                0425   740 ;       generic names, and the PTD$V_UNIT bit will be set iff
                                0425   741 ;       MAX_UNIT is greater than 0.
                                0425   742 ;
                                0425   743 ; Inputs and Outputs for this routine are in the local storage
                                0425   744 ; block allocated at the beginning of DSX$ATTACH.  It also references
                                0425   745 ; both the parent Ptable (if any) and the current Ptable descriptor,
                                0425   746 ; through pointers stored in said local block.
                                0425   747 ;
                                0425   748 ;--
                                0425   749
                                0425   750 Get_Attributes:                                ;                                [17]
                00FC 8F   BB    0425   751          PushR   #^M<R2,R3,R4,R5,R6,R7>  ; Save some registers                [17]
                  FF AD   94    0429   752          ClrB    Name_Flags(Fp)         ; No flags set                      [17]
                  EB AD   94    042C   753          ClrB    Generic(Fp)            ; No enforced generic yet           [17]
                  FE AD   94    042F   754          ClrB    Parent_Controller(Fp)  ; No enforced controller yet        [17]
             57   F9 AD   D0    0432   755          MovL    PtDesc(Fp), R7         ; Get Ptable descriptor start       [17]
                50   87   9A    0436   756          MovZBL  (R7)+, R0              ; Get length of type string         [17]
            FD AD   01 A740   90    0439   757          MovB    1(R7)[R0], Max_Unit(Fp) ; Fetch maximum unit number       [17]
                     07   13    043F   758          BEqlU   10$                    ; Branch if old ATTACH said no unit [17]
                FF AD   01   90    0441   759          MovB    #Ptd$M_Unit, -         ; If there is no NAME statement,    [17]
                                0445   760                   Name_Flags(Fp)         ; .. this device must have unit     [17]
                   FD AD   97    0445   761          DecB    Max_Unit(Fp)           ; Convert count to max, not max+1   [19]
             57   04 A740   9E    0448   762 10$:     MovAB   4(R7)[R0], R7          ; Skip all of $DS_$INITIALIZE       [19]
                80 8F   87   91    044D   763          CmpB    (R7)+, #PD$_Start      ; Next byte ought to be start code  [17]
                     5B   12    0451   764          BNeq    20$                    ; Skip out if something's wrong     [17]
                8D 8F   87   91    0453   765          CmpB    (R7)+, #PD$_Name       ; Is the next byte start of NAME?   [17]
                     55   12    0457   766          BNeq    20$                    ; Skip out if it's not              [17]
                FF AD   87   90    0459   767          MovB    (R7)+, Name_Flags(Fp)  ; Set real name flags               [17]
                50   87   9A    045D   768          MovZBL  (R7)+, R0              ; Get length of generic name        [17]
                EB AD   50   90    0460   769          MovB    R0, Generic(Fp)        ; Save length away                  [17]
EC AD   09   00   67   50   2C    0464   770          MovC5   R0, (R7), -            ; Copy generic string               [17]
                                046B   771                   #0, -                  ; .. zero filled                    [17]
                                046B   772                   #9, Generic+1(Fp)      ; .. into local storage             [17]
                50   F5 AD   D0    046B   773          MovL    Parent_Ptable(Fp), R0  ; Get parent Ptable                 [17]
                     3D   13    046F   774          BEql    20$                    ; Exit if attached to HUB           [17]
                FF AD   18   93    0471   775          BitB    #Ptd$M_Inherit, -      ; Check if device should inherit either [19]
                                0475   776                   Name_Flags(Fp)         ; .. name or controller             [19]
                     37   13    0475   777          BEql    20$                    ; Skip if niether bit is set        [19]
             55   04 A0   01   C1    0477   778          AddL3   #1, Hp$Q_Device+4(R0), -; Get address of parent name (- '_')  [17]
                                047C   779                   R5                     ;                                   [17]
                54   60   01   A3    047C   780          SubW3   #1, Hp$Q_Device(R0), R4 ; Get length of same               [17]
                     54   54   3C    0480   781          MovZWL  R4, R4                 ; Zero extend length                [17]
                 00000000'EF   16    0483   782          Jsb     L^Scan$Alpha           ; Scan over alphabetic              [17]
                     23   13    0489   783          BEql    20$                    ; Exit if none there (??)           [17]
                52   50   01   C3    048B   784          SubL3   #1, R0, R2             ; Get length minus 1                [17]
                     02   52   D1    048F   785          CmpL    R2, #2                 ; Generic should be at least 2 chars [17]
```

```
                                 1A   1F  0492   786          BLssU   20$                             ; Don't use if not valid generic form
                      05 FF AD   04   E1  0494   787          BbC     #Ptd$V_Inherit_Con, -           ; Don't use controller letter if
                                                0499   788                   Name_Flags(Fp), 15$     ; .. not supposed to
                       FE AD   6142   90  0499   789          MovB    (R1)[R2], -                     ; Store parent's controller letter
                                                049E   790                   Parent_Controller(Fp)   ;
                      0B FF AD   03   E1  049E   791 15$:     BbC     #Ptd$V_Inherit_Pre, -           ; Don't use prefix if shouldn't
                                                04A3   792                   Name_Flags(Fp), 20$     ; .. inherit it
                         EB AD   52   90  04A3   793          MovB    R2, Generic(Fp)                 ; If valid, use parent name as generic
                EC AD   09   00   61   52   2C  04A7   794          MovC5   R2, (R1), -             ; Copy to Generic, fill with 0
                                                04AE   795                   #0, -                   ;
                                                04AE   796                   #9, Generic+1(Fp)       ;
                          00000000'EF   E9  04AE   797 20$:     BlbC    L^Ds$GB_Inhibit_Naming,-        ; If should not enforce naming
                                   0B       04B4   798                   40$                     ;
                                EB AD   94  04B5   799 30$:     ClrB    Generic(Fp)                     ; .. then no enforced generics
                                FE AD   94  04B8   800          ClrB    Parent_Controller(Fp)           ; .. or controller
                      FF AD     FA 8F   8A  04BB   801          BicB2   #^C<Ptd$M_Unit ! -              ; .. and allow only unit and name
                                                04C0   802                   Ptd$M_Name>, -           ; .... flags
                                                04C0   803                   Name_Flags(Fp)          ;
                                00FC 8F   BA  04C0   804 40$:     PopR    #^M<R2,R3,R4,R5,R6,R7>          ; Restore registers
                                          05  04C4   805          Rsb                                     ; Return
                                              04C5   806
```

```
                        04C5   808 .sbttl  parse device name
                        04C5   809 ;
                        04C5   810 ; Small routine to save hassle in checking for alphas
                        04C5   811 ;
                        04C5   812
                        04C5   813 if_alpha:                                    ;                                       [17]
                  50    D4     04C5   814         clrl    r0                     ; Assume bad                            [17]
                  54    D5     04C7   815         tstl    r4                     ; Any string left?                      [17]
                  0F    15     04C9   816         bleq    10$                    ; Branch if not                         [17]
           41 8F  65    91     04CB   817         cmpb    (r5), #^A"A"           ; Check for alpha                       [17]
                  08    19     04CF   818         blss    10$                    ; Branch if not                         [17]
           5A 8F  65    91     04D1   819         cmpb    (r5), #^A"Z"           ;                                       [17]
                  02    14     04D5   820         bgtr    10$                    ; Branch if not                         [17]
                  50    D6     04D7   821         incl    r0                     ; It's OK                               [17]
                        05     04D9   822 10$:     rsb                            ; Return                                [17]
                               04DA   823
                               04DA   824 ;
                               04DA   825 ; This routine parses a device name.  It's output is the same as that for
                               04DA   826 ; the SCAN$DEVICE routine in PARSE.  However, it takes into account the
                               04DA   827 ; generic name and flags of the NAME statement in parsing. This routine
                               04DA   828 ; can handle device names of the form 'node$ggan'.
                               04DA   829 ;
                               04DA   830 ; NOTE: This code assumes that if the PTD$V_INHERITED flag is set, the link
                               04DA   831 ;       device will have a controller letter.  If there is ever a chain of
                               04DA   832 ;       devices for which this might not be true, the code must be changed.
                               04DA   833 ;
                               04DA   834 ; Inputs:
                               04DA   835 ;       R4 => remaining length of input string
                               04DA   836 ;       R5 => address of remaining input string
                               04DA   837 ;
                               04DA   838 ; Outputs:
                               04DA   839 ;       R0 => low word length of device name which was parsed (0 if none)
                               04DA   840 ;             high word has device unit number (-1 if none)
                               04DA   841 ;       R1 => address of device name (input R5)
                               04DA   842 ;       R2 => Success code:
                               04DA   843 ;             0 = OK (no errors found: may not be any name at all)
                               04DA   844 ;             1 = error in generic prefix
                               04DA   845 ;             2 = error in controller
                               04DA   846 ;             3 = error in unit number
                               04DA   847 ;             4 = error in node name format
                               04DA   848 ;       R4 => updated past device name (if any)
                               04DA   849 ;       R5 => updated past device name (if any)
                               04DA   850 ;
                               04DA   851
                               04DA   852 parse_device::                           ;                                       [17]
              00FF 8F  BB      04DA   853         pushr   #^M<r0,r1,r2,r3,r4,r5,r6,r7>    ; Save registers                [17]
        57      FF AD  9A      04DE   854         MovZBL  Name_Flags(Fp), R7     ; Get flags conveniently local          [17]
        04 AE    55 D0  04E2   855         movl    r5, 4(sp)                      ; Set saved R1 to start of string       [17]
        58 57    02 E0  04E6   856         bbs     #ptd$v_name, R7, 40$           ; Handle differently if it's node name  [17]
           50    54 D0  04EA   857         MOVL    R4,R0                          ; use for index                         [28]
                        04ED   858 1$:
           24    85 91  04ED   859         CMPB    (R5)+,#^A"$"                   ; check for new device name format      [28]
                 09 13  04F0   860         BEQL    2$                             ; if new format then 2$                 [28]
              F8 50 F5  04F2   861         SOBGTR  R0,1$                          ; do for all name length                [28]
        55    04 AE D0  04F5   862         MOVL    4(SP),R5                       ; restore input string pointer          [28]
                 12 11  04F9   863         BRB     5$                             ; continue as before                    [28]
                        04FB   864 2$:
```

```
                0A AB   04    88  04FB   865          BISB2   #HP$M_NAME,HP$B_FLAGS(R11) ; we have a long device name        [28]
                55   04 AE    D0  04FF   866          MOVL    4(SP),R5                   ; restore input string pointer       [28]
                08 AE   04    D0  0503   867          MOVL    #4,8(SP)                   ; condition error code               [28]
                   53   0A    D0  0507   868          MOVL    #10,R3                     ; max length for'node'of 'node$ggan'name
                      00B2    31  050A   869          BRW     122$                       ; go to 122$                         [28]
                              050D   870 5$:
                08 AE   01    D0  050D   871          movl    #1, 8(sp)                  ; Assume prefix error in saved R2    [17]
                      52      D4  0511   872          clrl    r2                         ; Flag for un-enforced prefix devices [17]
                51   EB AD    9E  0513   873 10$:      MovAB   Generic(Fp), R1            ; Get address of enforced generic    [17]
                   50   81    9A  0517   874          MovZBL  (R1)+, R0                  ; .. and it's length                 [17]
                         03    12  051A   875          bneq    20$                        ; Branch if OK to check              [17]
                            0088 31  051C   876          brw     110$                       ; No enforced gereric for this device [17]
                         54    D5  051F   877 20$:      tstl    r4                         ; Any string left ?                  [17]
                         7D    15  0521   878          bleq    100$                       ; Error if none                      [17]
                   85    81    91  0523   879          cmpb    (r1)+, (r5)+               ; Compare a byte                     [17]
                         78    12  0526   880          bneq    100$                       ; No good if mismatch                [17]
                         54    D7  0528   881          decl    r4                         ; Count the byte we checked          [17]
                      F2 50    F5  052A   882          subgtr  r0, 20$                    ; And loop til we're done            [17]
                      08 AE    D6  052D   883 30$:      incl    8(sp)                      ; Increment saved R2                 [17]
                   57   01    E0  0530   884          bbs     #ptd$v_controller, -       ; Is there supposed to be a controller? [17]
                         11          0533   885                   R7, 50$               ; ...if so, check for it             [17]
                         8F    10  0534   886          bsbb    if_alpha                   ; Is next character an alpha?        [17]
                      22 50    E9  0536   887          blbc    r0, 60$                    ; If not, then we're OK--check unit  [17]
                      64 52    E9  0539   888          blbc    r2, 100$                   ; If enforced prefix, that's an error [17]
                         54    D7  053C   889          decl    r4                         ; Skip over the backtracked byte     [17]
                         55    D6  053E   890          incl    r5                         ;                                    [17]
                         19    11  0540   891          brb     60$                        ; Check unit number                  [17]
                      0073     31  0542   892 40$:      brw     120$                       ; Just a step for a bbs              [17]
                              0545   893 50$:                                           ; Need controller letter             [17]
                      FF7D    30  0545   894          bsbW    if_alpha                   ; Check next character               [17]
                   55 50      E9  0548   895          blbc    r0, 100$                   ; Error if not a letter!             [17]
                         54    D7  054B   896          decl    r4                         ; Want to lose this                  [17]
                   50    85    9A  054D   897          movzbl  (r5)+, r0                  ; Get controller letter              [17]
                56   FE AD    9A  0550   898          MovZBL  Parent_Controller(Fp), -   ; Get correct letter              [17]
                              0554   899                   R6                         ; .. into R6                         [17]
                         05    13  0554   900          beql    60$                        ; Not specified, check unit          [17]
                   56    50    D1  0556   901          cmpl    r0, r6                     ; Compare them                       [17]
                         45    12  0559   902          bneq    100$                       ; Error if not equal                 [17]
                              055B   903 60$:                                           ; Check for unit                     [17]
                      08 AE    D6  055B   904          incl    8(sp)                      ; Increment assumed error code       [17]
                00000000'EF    16  055E   905          Jsb     L^scan$decimal             ; Is there a number here?            [17]
                         09    12  0564   906          bneq    70$                        ; Branch if was                      [17]
                36 57    00    E0  0566   907          bbs     #ptd$v_unit, R7, 100$      ; Error if wasn't, and needed one    [17]
                   50    01    CE  056A   908          mnegl   #1, r0                     ; Set no unit number code in R0      [17]
                         04    11  056D   909          brb     80$                        ; Join common                        [17]
                              056F   910 70$:                                           ; Found a unit number                [17]
                2D 57    00    E1  056F   911          bbc     #ptd$v_unit, R7, 100$      ; Error if was, and didn't want one  [17]
                02 AE    50    B0  0573   912 80$:      movw    r0, 2(sp)                  ; Set high word of saved R0          [17]
             6E 55    14 AE    A3  0577   913          subw3   20(sp), r5, (sp)           ; Calculate length of name found     [17]
                         54    D5  057C   914          tstl    r4                         ; Check for more characters          [17]
                         13    15  057E   915          bleq    90$                        ; Really OK either way...            [17]
                         54    D7  0580   916          decl    r4                         ; Assume we'll want next character   [17]
                   3A    85    91  0582   917          cmpb    (r5)+, #^A":"              ; Is it a colon?                     [17]
                         0C    13  0585   918          beql    90$                        ; That's cool, a colon feels just fine [17]
                         54    D6  0587   919          incl    r4                         ; Don't want next character, elsewise [17]
                   20    75    91  0589   920          cmpb    -(r5), #^A" "              ; Is next a space?                   [17]
                         05    13  058C   921          beql    90$                        ; Can be terminated by space         [17]
```

```
                09   65   91   058E   922          cmpb    (r5), #^x9                      ; Is it a tab?                                          [17]
                     0D   12   0591   923          bneq    100$                            ; Otherwise, may be hex 'unit number'                   [17]
                               0593   924 90$:                                            ; Here's the successful return stuff                    [17]
             10 AE   54   7D   0593   925          movq    r4, 16(sp)                      ; Update saved pointers                                 [17]
                00FF 8F   BA   0597   926          popr    #^M<r0,r1,r2,r3,r4,r5,r6,r7>    ; Restore the registers                                 [17]
                     52   D4   059B   927          clrl    r2                              ; No error found                                        [17]
                     50   B5   059D   928          tstw    r0                              ; Check for anything found                              [17]
                          05   059F   929          rsb                                     ; And return happily                                    [17]
                               05A0   930 100$:                                           ; Error return                                          [17]
                00FF 8F   BA   05A0   931          popr    #^M<r0,r1,r2,r3,r4,r5,r6,r7>    ; Restore registers                                     [17]
                     50   7C   05A4   932          clrq    r0                              ; Act as if nothing found                               [17]
                          05   05A6   933          rsb                                     ; and return                                            [17]
                               05A7   934 110$:                                           ; Here if no enforced prefix                            [17]
                53   02   D0   05A7   935          movl    #2, r3                          ; Set min length for generic                            [17]
                     4E   10   05AA   936          bsbb    scan_alpha                      ; Check string                                          [17]
                     F2   19   05AC   937          blss    100$                            ; If too few, error                                     [17]
                     55   D7   05AE   938          decl    r5                              ; Backup for controller check                           [17]
                     54   D6   05B0   939          incl    r4                              ;                                                       [17]
                52   01   D0   05B2   940          movl    #1, r2                          ; set flag                                              [17]
                   FF75   31   05B5   941          brw     30$                             ; Check controller                                      [17]
                               05B8   942 120$:                                           ; Node name check                                       [17]
             08 AE   04   D0   05B8   943          movl    #4, 8(sp)                       ; Give back node name error code                        [17]
                53   06   D0   05BC   944          movl    #6, r3                          ; Maximum length of node name                           [17]
                               05BF   945 122$:
                     45   10   05BF   946          bsbb    scan_alphanum                   ; Check it                                              [24]
                     DD   1A   05C1   947          BGtrU   100$                            ; Branch if error                                       [17]
                     50   B5   05C3   948          TstW    R0                              ; Check for zero length, too                            [17]
                     D9   15   05C5   949          BLeq    100$                            ; Error if nothing                                      [17]
                57   03   D3   05C7   950          BITL    #PTD$M_CONTROLLER!PTD$M_UNIT, R7 ; Check for more field proc'g                          [26]
                     27   13   05CA   951          BEQL    135$                            ; Branch if not                                         [26]
                     54   D5   05CC   952          TSTL    R4                              ; Anything left ?                                       [26]
                     1E   15   05CE   953          Bleq    129$                            ; Branch if not                                         [26]
                24   65   91   05D0   954          CMPB    (R5),#^A"$"                     ; should be '$'                                         [28]
                     1E   12   05D3   955          BNEQ    135$                            ; Branch if not.                                        [26]
                     55   D6   05D5   956          INCL    R5                              ; Skip '$'.                                             [26]
                     54   D7   05D7   957          DECL    R4                              ; Skip character                                        [26]
                     34   BB   05D9   958          PUSHR   #^M<R2,R4,R5>                   ; save registers                                        [28]
          52   0C   AB   9E   05DB   959          MOVAB   HP$T_DEVICE(R11),R2             ; pointer to device name                                [28]
          82   5F   8F   90   05DF   960          MOVB    #^A"_",(R2)+                    ; insert '_' physical device indicator                  [28]
                               05E3   961 128$:
                20   65   91   05E3   962          CMPB    (R5),#^A" "                     ; are we finished ?                                     [28]
                     06   13   05E6   963          BEQL    129$                            ; then go to 129$                                       [28]
                82   85   90   05E8   964          MOVB    (R5)+,(R2)+                     ; copy byte                                             [28]
                   F5 54   F5   05EB   965          SOBGTR  R4,128$                         ; do for whole string                                   [28]
                               05EE   966 129$:
                     34   BA   05EE   967          POPR    #^M<R2,R4,R5>                   ; restore registers                                     [28]
                      FF1A   31   05F0   968          BRW     5$                              ;Check for other enforced functions.                    [26]
          50   FF   8F   98   05F3   969 135$:    CvtBL   #-1, R0                         ; Set to return 'no unit number'                        [17]
                    FF79   31   05F7   970          BRW     80$                             ; OK if less than                                       [17]
                               05FA   971
                               05FA   972 ;
                               05FA   973 ; Small routine to call scan$alpha and compare return length
                               05FA   974 ;
                               05FA   975
                               05FA   976 scan_alpha:                                     ;                                                       [17]
                     08   BB   05FA   977          pushr   #^M<r3>                         ; Save the compare length                               [17]
00000000'EF                    16   05FC   978          Jsb     L^scan$alpha                    ; Call PARSE scan routine                               [17]
```

```
                8E    50    D1  0602   979          cmpl    r0, (sp)+                       ; Do comparison                                 [17]
                            05  0605   980          rsb                                     ; Return                                        [17]
                                0606   981 ;
                                0606   982 ; Small routine to call scan$alphanum and compare return length
                                0606   983 ;
                                0606   984
                                0606   985 scan_alphanum:                                   ;                                               [24]
                      08    BB  0606   986          pushr   #^M<r3>                         ; Save the compare length                       [24]
          00000000'EF       16  0608   987          Jsb     L^scan$alphanum                 ; Call PARSE scan routine                       [24]
                8E    50    D1  060E   988          cmpl    r0, (sp)+                       ; Do comparison                                 [24]
                            05  0611   989          rsb                                     ; Return                                        [24]
```

```
0612   991          .SBTTL  PT_INTERPRET    Decode PT-descriptor
0612   992
0612   993 ;++
0612   994 ; FUNCTIONAL DESCRIPTION:
0612   995 ;
0612   996 ;       This routine is used to interpret the P-table descriptor
0612   997 ;       which builds the P-table for the current unit.
0612   998 ;       Each OP code is examined and executed.  If during the
0612   999 ;       execution input text is required, the input string is examined
0612  1000 ;       if the string is empty the input routine is called to get input.
0612  1001 ;
0612  1002 ; CALLING SEQUENCE:
0612  1003 ;
0612  1004 ;       PT$INTERPRET (INITPC, BASE, TEXT, PROMPT)
C612  1005 ;
0612  1006 ; INPUT PARAMETERS:
0612  1007 ;
0612  1008 ;       INITPC  Initial PC for interpretation
0612  1009 ;       BASE    Base for data storage
0612  1010 ;       TEXT    Address of Quad descriptor of text
0612  1011 ;       PROMPT  Address of buffer to receive ASCIC prompt text
0612  1012 ;               if failure
0612  1013 ;
0612  1014 ; IMPLICIT INPUTS:      NONE
0612  1015 ;
0612  1016 ; OUTPUT PARAMETERS:
0612  1017 ;
0612  1018 ; IMPLICIT OUTPUTS:
0612  1019 ;
0612  1020 ;       P-table effected by intpretation of PT-descriptor
0612  1021 ;
0612  1022 ; REGISTER USAGE:
0612  1023 ;
0612  1024 ;       R0      Length of compare string/scratch
0612  1025 ;       R1      Address of compare string/scratch
0612  1026 ;       R2      Current index in PD_STRING/scratch
0612  1027 ;       R3      Remaining length in PD_STRING/scratch
0612  1028 ;       R4      Length of input string
0612  1029 ;       R5      Address of input string
0612  1030 ;       R6      Address of prompt string descriptor
0612  1031 ;       R8      Address of input descriptor
0612  1032 ;       R9      Current VALUE
0612  1033 ;       R10     Virtural PC for P-table intpretation
0612  1034 ;       R11     Base of P-table
0612  1035 ;
0612  1036 ; COMPLETION CODES:
0612  1037 ;
0612  1038 ;       SS$_NORMAL
0612  1039 ;       SS$_BADPARAM
0612  1040 ;       SS$_INSFARG
0612  1041 ;--
```

```
                     0612  1043
                     0612  1044          $OFFSET 4,POSITIVE,< -
                     0612  1045                  <IARG$_PC,4>,-
                     0612  1046                  <IARG$_BASE,4>,-
                     0612  1047                  <IARG$_TEXT,4>,-
                     0612  1048                  <IARG$_PROMPT,4>>
                     0004          IARG$_PC:
                     0008          IARG$_BASE:
                     000C          IARG$_TEXT:
                     0010          IARG$_PROMPT:
                     0612  1049
                     0612  1050          $OFFSET 0,NEGATIVE,< -
                     0612  1051                  <OLDPC,4>,-
                     0612  1052                  <DESC,8>,-
                     C612  1053                  <BUFFER,132>,-
                     0612  1054                  <LOCAL,0>>
                     FFFC          OLDPC:
                     FFF4          DESC:
                     FF70          BUFFER:
                     FF70          LOCAL:
                     0612  1055
```

```
                                0612  1057
                         OFFC   0612  1058 .ENTRY  PT$INTERPRET,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                0614  1059
      5E    FF70 CD    9E     0614  1060          MOVAB   LOCAL(FP),SP              ; Allocate space for local storage
   F8 AD    FF70 CD    9E     0619  1061          MOVAB   BUFFER(FP),DESC+4(FP)     ; Address of buffer to descriptor
                              061F  1062
      58     0C BC     7E     061F  1063          MOVAQ   @IARG$_TEXT(AP),R8        ; Get address of text descriptor
         54     68     7D     0623  1064          MOVQ    (R8),R4                   ; Fetch descriptor
         54     54     3C     0626  1065          MOVZWL  R4,R4                     ; Be sure we only use word of length
      56     10 BC     7E     0629  1066          MOVAQ   @IARG$_PROMPT(AP),R6      ; Set address of prompt string
      5A     04 AC     D0     062D  1067          MOVL    IARG$_PC(AP),R10          ; Get initial PC
      5B     08 AC     D0     0631  1068          MOVL    IARG$_BASE(AP),R11        ; Base of P-table
      6A     80 8F     91     0635  1069          CMPB    #PD$_START,(R10)          ; Must start with start OP
                 25    12     0639  1070          BNEQ    BADPARAM                  ; Branch if not
                              063B  1071 INTERPRET:
      FC AD     5A     D0     063B  1072          MOVL    R10,OLDPC(FP)             ; Incase of backup
                              063F  1073          CASE    SRC=(R10)+,TYPE=B,LIMIT=#PD$_START,DISPLIST=< -
                              063F  1074                  PD_START, -
                              063F  1075                  PD_FINISH, -
                              063F  1076                  PD_DECIMAL, -
                              063F  1077                  PD_OCTAL, -
                              063F  1078                  PD_HEXADECIMAL, -
                              063F  1079                  PD_STRING, -
                              063F  1080                  PD_LITERAL, -
                              063F  1081                  PD_FETCH, -
                              063F  1082                  PD_STORE, -
                              063F  1083                  PD_COMPLEMENT, -
                              063F  1084                  pd_add, -                 ; Add statment                    [09]
                              063F  1085                  pd_logical, -             ; Logical prompt statment         [09]
                              063F  1086                  pd_case, -                ; Case statement                  [09]
                              063F  1087                  pd_name>                  ; Name directive                  [17]
                              0660  1088 BADPARAM:
         50    14    D0     0660  1089          MOVL    S^#SS$_BADPARAM,R0        ; Bad PT-desc
                     04     0663  1090          RET                               ; Return
                              0664  1091
                              0664  1092 PD_START:
         59    01    CE     0664  1093          MNEGL   #1,R9                     ; Initialize to minus 1
               D2    11     0667  1094          BRB     INTERPRET                 ; Next
                              0669  1095
                              0669  1096 PD_FINISH:
         50    01    D0     0669  1097          MOVL    S^#SS$_NORMAL,R0          ; It worked
                     04     066C  1098          RET
                              066D  1099
                              066D  1100 PD_LITERAL:
         59    8A    D0     066D  1101          MOVL    (R10)+,R9                 ; Set current value
               C9    11     0670  1102          BRB     INTERPRET                 ; Next!
                              0672  1103
                              0672  1104 PD_FETCH:
         50    8A    3C     0672  1105          MOVZWL  (R10)+,R0                 ; Get byte offset
         51    8A    9A     0675  1106          MOVZBL  (R10)+,R1                 ; Bit offset
         52    8A    9A     0678  1107          MOVZBL  (R10)+,R2                 ; Field size
   59 6B40    52    51 EF    067B  1108          EXTZV   R1,R2,(R11)[R0],R9        ; Get field
               B8    11     0681  1109          BRB     INTERPRET                 ; Next!
                              0683  1110
                              0683  1111 PD_STORE:
         50    8A    3C     0683  1112          MOVZWL  (R10)+,R0                 ; Get byte offset
         51    8A    9A     0686  1113          MOVZBL  (R10)+,R1                 ; Bit offset
```

```
                    52   8A   9A  0689  1114          MOVZBL  (R10)+,R2                ; Field size
          6B40      52   51   59  F0  068C  1115      INSV    R9,R1,R2,(R11)[R0]       ; Insert the field
                         A7   11  0692  1116          BRB     INTERPRET
                                  0694  1117
                                  0694  1118 PD_COMPLEMENT:
                    59   59   D2  0694  1119          MCOML   R9,R9                    ; Complement value
                         A2   11  0697  1120          BRB     INTERPRET                ; Next function
                                  0699  1121
                                  0699  1122 PD_ADD:
                    50   8A   3C  0699  1123          MOVZWL  (R10)+,R0                ; Get byte offset
                    51   8A   9A  069C  1124          MOVZBL  (R10)+,R1                ; Get bit offset
                    52   8A   9A  069F  1125          MOVZBL  (R10)+,R2                ; Field size
          53  6B40      52   51  EE  06A2  1126      EXTV    R1,R2,(R11)[R0],R3       ; Get sign extended field value
                    53   59   C0  06A8  1127          ADDL    R9,R3                    ; Add in current value
          6B40      52   51   53  F0  06AB  1128      INSV    R3,R1,R2,(R11)[R0]       ; Put field back
                         88   11  06B1  1129          BRB     INTERPRET                ; Next function
                                  06B3  1130
                                  06B3  1131 pd_logical:                              ; Process LOGICAL                [09]
                    52   5E   D0  06B3  1132          movl    sp, r2                   ; Save stack pointer             [09]
                    53   5A   D0  06B6  1133          movl    r10, r3                  ; Get address of prompt string   [09]
              00000A4D'EF        16  06B9  1134          Jsb     L^get_line               ; Get data                       [18]
                    68   50   E9  06BF  1135          blbc    r0, 50$                  ; Exit immediately if error      [09]
              00000000'EF        16  06C2  1136          Jsb     L^scan$alpha             ; Read alpha string              [18]
                         41   13  06C8  1137          beql    40$                      ; Error if none                  [09]
                    53   50   D0  06CA  1138          movl    r0, r3                   ; Save length read               [09]
                         59   D4  06CD  1139          clrl    r9                       ; Assume 'no'                    [09]
                7E   4F4E 8F   B0  06CF  1140          movw    #^A'NO', -(sp)           ; Push on 'no' to check          [09]
                     7E   02   90  06D4  1141          movb    #2, -(sp)                ; Make it ASCIC                  [09]
                      4E 8F   61   91  06D7  1142      cmpb    (r1), #^A'N'             ; Does it really start with an 'n'? [09]
                              0B   13  06DB  1143          beql    10$                      ; Yep--verify                    [09]
                         5E   52   D0  06DD  1144          movl    r2, sp                   ; Restore stack                  [09]
                              59   D6  06E0  1145          incl    r9                       ; It's a 'yes' or nothing        [09]
                     53455903 8F   DD  06E2  1146          pushl   #3+<^A'YES'@8>           ; Push on ASCIC 'yes'            [09]
                         50   8E   9A  06E8  1147 10$:     movzbl  (sp)+, r0                ; Get the length                 [09]
                         50   53   D1  06EB  1148          cmpl    r3, r0                   ; Is input too long?             [09]
                              1B   14  06EE  1149          bgtr    40$                      ; If so, error                   [09]
                         50   53   D0  06F0  1150          movl    r3, r0                   ; Copy to destructable it        [09]
                         8E   81   91  06F3  1151 20$:     cmpb    (r1)+, (sp)+             ; Compare bytes                  [09]
                              13   12  06F6  1152          bneq    40$                      ; Error!                         [09]
                            F8 50   F5  06F8  1153          sobgtr  r0, 20$                  ; Loop on the check              [09]
                         5E   52   D0  06FB  1154 30$:     movl    r2, sp                   ; Restore stack                  [09]
                         50   8A   9A  06FE  1155          movzbl  (r10)+, r0               ; Get length of prompt           [09]
                         5A   50   C0  0701  1156          addl2   r0, r10                  ; Skip it                        [09]
                      02 A8   53   A0  0704  1157          addw    r3, 2(r8)                ; We've just accepted more of string [09]
                            FF30   31  0708  1158          brw     interpret                ; Back for next statement        [09]
                         5E   52   D0  070B  1159 40$:     movl    r2, sp                   ; Restore stack                  [09]
                              6A   9F  070E  1160          pushab  (r10)                    ; Address of prompt              [09]
              00000273'EF        7F  0710  1161          pushaq  t_yesno                  ; 'yes,no'                       [09]
              00000269'EF        9F  0716  1162          pushab  t_shouldbe               ; 'Should be'                    [09]
              00000281'EF        7F  071C  1163          pushaq  t_string                 ; Use invalid string argument    [09]
            0A75'CF      04   FB  0722  1164          calls   #4, w^format_prompt      ; Format prompt return           [09]
                         50   14   D0  0727  1165          movl    #ss$_badparam, r0        ; Invalid parameter              [09]
                              04   7A  072A  1166 50$:     ret                              ;                                [09]
                                  072B  1167
                                  072B  1168 pd_case:                                 ; Case operation                 [09]
                         50   8A   9A  072B  1169          movzbl  (r10)+, r0               ; Count of cases                 [09]
                              0B   13  072E  1170          beql    20$                      ; Exit if none (it's OK, though) [09]
```

```
                        51   8A   7D  0730  1171 10$:     movq    (r10)+, r1              ; Get next case pair                    [09]
                        59   51   D1  0733  1172          cmpl    r1, r9                  ; Does this one match?                  [09]
                             06   13  0736  1173          beql    30$                     ; If it does, do replacement            [09]
                           F5 50  F5  0738  1174          sobgtr  r0, 10$                 ; Loop through cases                    [09]
                            FEFD  31  073B  1175 20$:     brw     interpret               ; OK, return                            [09]
                        59   52   D0  073E  1176 30$:     movl    r2, r9                  ; Do value replacement                  [09]
                    5A  F8 AA40   7E  0741  1177          movaq   -8(r10)[r0], r10        ; Skip rest of the values               [09]
                           F3   11  0746  1178          brb     20$                     ; Mission accomplished, as ordered      [09]
                                   0748  1179
                                   0748  1180 Pd_Name:                                  ; Just skip over the name directive     [17]
                      50   01 AA   9A  0748  1181          MovZBL  1(R10), R0              ; Get length of enforced generic        [17]
                    5A  02 AA40   9E  074C  1182          MovAB   2(R10)[R0], R10         ; Skip over it all                      [17]
                           FEE7   31  0751  1183          BrW     Interpret               ; Return                                [17]
                                   0754  1184
                                   0754  1185 PD_DECIMAL:
                        53   0A   D0  0754  1186          MOVL    #10,R3                  ; Set radix
                             008F 30  0757  1187          BSBW    PD_NUMERIC              ; Join common
41 21 20 43 41 21 00000762'010E0000'  075A  1188          .ASCID  "!AC !AC decimal !UL thru !UL!/!AC? "
55 21 20 6C 61 6D 69 63 65 64 20 43   0768
2F 21 4C 55 21 20 75 72 68 74 20 4C   0774
                  20 3F 43 41 21      0780
                                      0785  1189 PD_OCTAL:
                        53   08   D0  0785  1190          MOVL    #8,R3                   ; Set radix
                             5F   10  0788  1191          BSBB    PD_NUMERIC              ; Join common
41 21 20 43 41 21 00000792'010E0000'  078A  1192          .ASCID  "!AC !AC octal !6OL thru !6OL!/!AC? "
4C 4F 36 21 20 6C 61 74 63 6F 20 43   0798
2F 21 4C 4F 36 21 20 75 72 68 74 20   07A4
                  20 3F 43 41 21      07B0
                                      07B5  1193 PD_HEXADECIMAL:
                        53   10   D0  07B5  1194          MOVL    #16,R3                  ; Set radix
                             2F   10  07B8  1195          BSBB    PD_NUMERIC              ; join common
41 21 20 43 41 21 000007C2'010E0000'  07BA  1196          .ASCID  "!AC !AC hexadecimal !XL thru !XL!/!AC? "
61 6D 69 63 65 64 61 78 65 68 20 43   07C8
21 20 75 72 68 74 20 4C 58 21 20 6C   07D4
         20 3F 43 41 21 2F 21 4C 58   07E0
                                      07E9  1197 PD_NUMERIC:
                             53   DD  07E9  1198          PUSHL   R3                      ; Save radix
                        53   5A   D0  07EB  1199          MOVL    R10,R3                  ; Address of ascic prompt
                        52   5A   D0  07EE  1200          MOVL    R10,R2                  ; Save prompt string for later
                00000A4D'EF   16  07F1  1201          Jsb     L^GET_LINE              ; Span spaces, new input if necessary   [18]
                             53 8ED0  07F7  1202          POPL    R3                      ; restore radix
                          53 50   E9  07FA  1203          BLBC    R0,30$                  ; Return if input errors
                        51   8A   9A  07FD  1204          MOVZBL  (R10)+,R1               ; Length of prompt
                        5A   51   C0  0800  1205          ADDL    R1,R10                  ; Skip over prompt
                        57   55   D0  0803  1206          MOVL    R5,R7                   ; Save current string address
                             04   BB  0806  1207          PUSHR   #^M<R2>                 ; Save pointer on stack now
                00000000'EF   16  0808  1208          Jsb     L^SCAN$NUMERIC          ; Scan and fetch input                  [18]
                             04   BA  080E  1209          POPR    #^M<R2>                 ; Restore pointer
                             22   13  0810  1210          BEQL    20$                     ; Branch on bad input
                    57   55   57   C3  0812  1211          SUBL3   R7,R5,R7                ; Calculate length of number
                             51   D5  0816  1212          TSTL    R1                      ; High order portion must be zero
                             1A   12  0818  1213          BNEQ    20$                     ; Branch on out of range
                        6A   50   D1  081A  1214          CMPL    R0,(R10)                ; Compare data with low limit
                             15   1F  081D  1215          BLSSU   20$                     ; Branch if less than limit
                      04 AA  50   D1  081F  1216          CMPL    R0,4(R10)               ; Compare data with high limit
                             0F   1A  0823  1217          BGTRU   20$                     ; Branch if out of range
                        59   50   D0  0825  1218          MOVL    R0,R9                   ; Set current value
```

```
                    5A    08    C0  0828  1219          ADDL    #8,R10                  ; Skip over low/high
                          8E    D5  082B  1220          TSTL    (SP)+                   ; Remove edit string address
                 02 A8    57    A0  082D  1221          ADDW2   R7,2(R8)                ; Increase length used so far
                     FE07    31    0831  1222          BRW     INTERPRET               ; Next instruction
                                   0834  1223 20$:
                    50    8E    D0  0834  1224          MOVL    (SP)+,R0                ; Get address of error edit string
                          62    9F  0837  1225          PUSHAB  (R2)                    ; Prompt address
                          7E    6A  7D  0839  1226      MOVQ    (R10),-(SP)             ; Push lower/upper bounds
                00000269'EF     9F  083C  1227          PUSHAB  L^T_SHOULDBE            ; Descriptive text argument             [10]
                          62    9F  0842  1228          PUSHAB  (R2)                    ; Prompt
                          60    7F  0844  1229          PUSHAQ  (R0)                    ; Address of format descriptor
            00000A75'EF   06    FB  0846  1230          CALLS   #6,L^FORMAT_PROMPT      ; Do formatting & copy                  [10]
                    50    14    D0  084D  1231          MOVL    #SS$_BADPARAM,R0        ; Bad parameter return
                                   0850  1232 30$:
                                04  0850  1233          RET                             ; Return on error
```

```
                                0851  1235
                                0851  1236 PD_STRING:
          53   5A   D0   0851  1237          MOVL    R10,R3                  ; Copy address of ascic prompt
          57   53   D0   0854  1238          MOVL    R3,R7                   ; Save prompt address
 00000A4D'EF   16   0857  1239          Jsb     L^GET_LINE              ; Verify something there
       01 50   E8   085D  1240          BLBS    R0,10$                  ; Branch if it worked
                   04   0860  1241          RET                             ; Return if input spaz
                        0861  1242 10$:
          59   01   CE   0861  1243          MNEGL   #1,R9                   ; Initialize VALUE
          53   8A   9A   0864  1244          MOVZBL  (R10)+,R3               ; Get length of prompt
                        0867  1245
          5A   53   C0   0867  1246 20$:     ADDL    R3,R10                  ; Skip over something
          53   8A   9A   086A  1247          MOVZBL  (R10)+,R3               ; Get length of next something
                2D   13   086D  1248          BEQL    70$                     ; Branch if end of list
                59   D6   086F  1249          INCL    R9                      ; Update string index
                        0871  1250
    50    54   53   C3   0871  1251          SUBL3   R3,R4,R0                ; Length remaining after this match
                F0   19   0875  1252          BLSS    20$                     ; Input shorter, try next string
          51   55   D0   0877  1253          MOVL    R5,R1                   ; Copy input address
                05   11   087A  1254          BRB     40$                     ; Start with count
          8A   81   91   087C  1255 30$:     CMPB    (R1)+,(R10)+            ; Compare character
                E6   12   087F  1256          BNEQ    20$                     ; Branch if mismatch
            F8 53   F4   0881  1257 40$:     SOBGEQ  R3,30$                  ; Count and try again if more
    53    51   55   C3   0884  1258          SUBL3   R5,R1,R3                ; Calc length of string
      02  A8   53   A0   0888  1259          ADDW2   R3,2(R8)                ; Increase length used so far
          54   50   7D   088C  1260          MOVQ    R0,R4                   ; Put updated input pointers back
                53   D4   088F  1261          CLRL    R3                      ; Already in position, add zero
          5A   53   C0   0891  1262 50$:     ADDL    R3,R10                  ; Skip over this one
          53   8A   9A   0894  1263          MOVZBL  (R10)+,R3               ; Get length of next item
                F8   12   0897  1264          BNEQ    50$                     ; Branch if not end
             FD9F   31   0899  1265          BRW     INTERPRET               ; Next instruction
                        089C  1266
                        089C  1267 70$:
       52   FF70  CD   9E   089C  1268          MOVAB   BUFFER(FP),R2           ; Address buffer
                        08A1  1269
    5A   FC  AD   01   C1   08A1  1270          ADDL3   #1,OLDPC(FP),R10        ; Point to prompt
             50   8A   9A   08A6  1271          MOVZBL  (R10)+,R0               ; Length of prompt
             5A   50   C0   08A9  1272          ADDL    R0,R10                  ; Skip prompt
                        08AC  1273
             50   8A   9A   08AC  1274 80$:     MOVZBL  (R10)+,R0               ; Get length of possible string
                  0B   13   08AF  1275          BEQL    100$                    ; Branch if last string
             82   8A   90   08B1  1276 90$:     MOVB    (R10)+,(R2)+            ; Copy byte
                FA 50   F5   08B4  1277          SOBGTR  R0,90$                  ; Branch if any left
             82   2C   90   08B7  1278          MOVB    #^A',',(R2)+            ; Insert ','
                  F0   11   08BA  1279          BRB     80$                     ; Try again
                        08BC  1280
                  52   D7   08BC  1281 100$:    DECL    R2                      ; Remove ','
 F4 AD   52   F8  AD   C3   08BE  1282          SUBL3   DESC+4(FP),R2,DESC(FP)  ; Length of prompt
                  67   9F   08C4  1283          PUSHAB  (R7)                    ; Address of prompt string
                F4 AD   7F   08C6  1284          PUSHAQ  DESC(FP)                ; Address of strings
          00000269'EF   9F   08C9  1285          PUSHAB  L^T_SHOULDBE            ; Address of "should be" insert         [10]
          00000281'EF   7F   08CF  1286          PUSHAQ  L^T_STRING              ; Address of format descriptor          [10]
    00000A75'EF   04   FB   08D5  1287          CALLS   #4,L^FORMAT_PROMPT      ; Do formatting                         [10]
             50   14   D0   08DC  1288          MOVL    #SS$_BADPARAM,R0        ; Bad parameter
                   04   08DF  1289          RET
```

```
                        08E0  1291          .SBTTL  DP$EXTRACT      Make P-table printable
                        08E0  1292
                        08E0  1293 ;++
                        08E0  1294 ; FUNCTIONAL DESCRIPTION:
                        08E0  1295 ;
                        08E0  1296 ;       This routine can be used to convert the data stored in
                        08E0  1297 ;       a P-table using the PT-descriptor to an ASCII string to be
                        08E0  1298 ;       printed.
                        08E0  1299 ;
                        08E0  1300 ; CALLING SEQUENCE:
                        08E0  1301 ;
                        08E0  1302 ;       PT$EXTRACT(PC,BASE,LENGTH,ADDRESS)
                        08E0  1303 ;
                        08E0  1304 ; INPUT PARAMETERS:
                        08E0  1305 ;
                        08E0  1306 ;       PC      Start of PT-descriptor
                        08E0  1307 ;       BASE    Base of P-table
                        08E0  1308 ;       TEXT    Address of QUAD descriptor of buffer
                        08E0  1309 ;
                        08E0  1310 ; IMPLICIT INPUTS:
                        08E0  1311 ;
                        08E0  1312 ; OUTPUT PARAMETERS:
                        08E0  1313 ;
                        08E0  1314 ;       TEXT    The length in the descriptor is updated
                        08E0  1315 ;
                        08E0  1316 ; IMPLICIT OUTPUTS:     NONE
                        08E0  1317 ;
                        08E0  1318 ; COMPLETION CODES:
                        08E0  1319 ;
                        08E0  1320 ;       SS$_BADPARAM    Incorrectly formed PT-descriptor
                        08E0  1321 ;       SS$_NORMAL      Completed successfully
                        08E0  1322 ;
                        08E0  1323 ; REGISTER USAGE:
                        08E0  1324 ;
                        08E0  1325 ;       R9      current VALUE
                        08E0  1326 ;       R10     Current PC in PT-descriptor
                        08E0  1327 ;       R11     Base address of P-table
                        08E0  1328 ;--
```

```
                                08E0  1330
                                08E0  1331          $OFFSET 4,POSITIVE,< -
                                08E0  1332          <EARG$_PC,4>, -
                                08E0  1333          <EARG$_BASE,4>, -
                                08E0  1334          <EARG$_TEXT,4>>
                                0004        EARG$_PC:
                                0008        EARG$_BASE:
                                000C        EARG$_TEXT:
                                08E0  1335
                          0FFC  08E0  1336 .ENTRY  PT$EXTRACT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                08E2  1337
                    5A  04 AC   7D  08E2  1338          MOVQ    EARG$_PC(AP),R10        ; Get PC and BASE
                    57  0C BC   7D  08E6  1339          MOVQ    @EARG$_TEXT(AP),R7      ; Get length of buffer
                        56      D4  08EA  1340          CLRL    R6                      ; Clear Last valid edit string
                    6A  80 8F   91  08EC  1341          CMPB    #PD$_START,(R10)        ; Verify good format
                        21      12  08F0  1342          BNEQ    EX_BADPARAM             ; WRONG!
                                08F2  1343 EXTRACT:
                                08F2  1344          CASE    SRC=(R10)+,TYPE=B,LIMIT=#PD$_START,DISPLIST=< -
                                08F2  1345                  EX_START, -
                                08F2  1346                  EX_FINISH, -
                                08F2  1347                  EX_DECIMAL, -
                                08F2  1348                  EX_OCTAL, -
                                08F2  1349                  EX_HEXADECIMAL, -
                                08F2  1350                  EX_STRING, -
                                08F2  1351                  EX_LITERAL, -
                                08F2  1352                  EX_FETCH, -
                                08F2  1353                  EX_STORE  -
                                08F2  1354                  EX_COMPLEMENT, -
                                08F2  1355                  ex_add, -               ; Add operation         [09]
                                08F2  1356                  ex_logical, -           ; logical               [09]
                                08F2  1357                  ex_case, -              ; Case                  [09]
                                08F2  1358                  ex_name>                ; name                  [17]
                                0913  1359 EX_BADPARAM:
                    50  14  D0  0913  1360          MOVL    S^#SS$_BADPARAM,R0      ; bad format PT-descriptor
                        04      0916  1361          RET
                                0917  1362
                                0917  1363 EX_START:
                        D9  11  0917  1364          BRB     EXTRACT                 ; Nop
                                0919  1365 EX_DECIMAL:
                                0919  1366 EX_OCTAL:
                                0919  1367 EX_HEXADECIMAL:
                56  FF AA    9E  0919  1368          MOVAB   -1(R10),R6              ; Save PC of prompt type inst
                        1F  10  091D  1369          BSBB    SKIP                    ; Skip prompt
                    5A  08  C0  091F  1370          ADDL    #8,R10                  ; Skip low and high limits
                        CE  11  0922  1371          BRB     EXTRACT                 ; Next inst
                                0924  1372 EX_STRING:
                56  FF AA    9E  0924  1373          MOVAB   -1(R10),R6              ; Save address of prompt type inst
                        13  10  0928  1374          BSBB    SKIP
                        50  D4  092A  1375          CLRL    R0                      ; Clear advance length
                                092C  1376 10$:
                    5A  50  C0  092C  1377          ADDL    R0,R10                  ; Skip something
                    50  8A  9A  092F  1378          MOVZBL  (R10)+,R0               ; Get length
                        F8  12  0932  1379          BNEQ    10$                     ; Try again
                        BC  11  0934  1380          BRB     EXTRACT                 ; Next instruction
                                0936  1381 ex_logical:                              ;                       [09]
                56  FF AA    9E  0936  1382          movab   -1(r10), r6             ; Save address of directive [09]
                    B5 AF    9F  093A  1383          pushab  extract                 ; Set up for SKIP to do RSB [09]
```

```
                                093D  1384 SKIP:
                          50    8A    9A 093D  1385          MOVZBL  (R10)+,R0               ; Get length
                          5A    50    C0 0940  1386          ADDL    R0,R10                  ; Skip string
                                      05 0943  1387          RSB
                                         0944  1388
                                         0944  1389 EX_LITERAL:
                                         0944  1390 EX_FETCH:
                                         0944  1391 EX_ADD:
                                      8A    D5 0944  1392          TSTL    (R10)+                  ; Skip literal or word+byte+byte
                                         0946  1393 EX_COMPLEMENT:
                                      AA    11 0946  1394          BRB     EXTRACT                 ; next!
                                         0948  1395 ex_case:                                    ;                                       [09]
                                50    8A    9A 0948  1396          movzbl  (r10)+, r0              ; Get number of cases                   [09]
                             5A    6A40    7E 094B  1397          movaq   (r10)[r0], r10          ; Skip the cases                        [09]
                                      A1    11 094F  1398          brb     extract                 ; And continue interpretation           [09]
                                         0951  1399
                                         0951  1400 Ex_Name:                                    ; Skip over Name directive              [17]
                          50    01 AA    9A 0951  1401          MovZBL  1(R10), R0              ; Get length of enforced generic        [17]
                       5A    02 AA40    9E 0955  1402          MovAB   2(R10)[R0], R10         ; Skip over it all                      [17]
                                      96    11 095A  1403          BrB     Extract                 ; Return                                [17]
                                         095C  1404
                                         095C  1405 EX_FINISH:
                          50    0C BC    7D 095C  1406          MOVQ    @EARG$_TEXT(AP),R0      ; Get length and address again
                    0C BC    58    51    C3 0960  1407          SUBL3   R1,R8,@EARG$_TEXT(AP)   ; Store new length
                                50    01    D0 0965  1408          MOVL    S^#SS$_NORMAL,R0        ; It worked
                                      04    04 0968  1409          RET
```

```
                                0969  1411
                                0969  1412 EX_STORE:
                          50    8A    3C  0969  1413          MOVZWL  (R10)+,R0               ; Get byte offset
                          51    8A    9A  096C  1414          MOVZBL  (R10)+,R1               ; Get bit offset
                          52    8A    9A  096F  1415          MOVZBL  (R10)+,R2               ; Get field size
                    59    6B40  52    51  EF  0972  1416          EXTZV   R1,R2,(R11)[R0],R9      ; Set current value
                                      56  D5  0978  1417          TSTL    R6                      ; Valid?
                                      03  12  097A  1418          BNeq    5$                      ; Yes, continue                 [17]
                                  FF73    31  097C  1419          BrW     EXTRACT                 ; No, next instruction          [17]
```

```
                                097F  1421 5$:     CASE    SRC=(R6)+,TYPE=B,LIMIT=#PD$_DECIMAL,DISPLIST=< -
                                097F  1422                  st_decimal,st_octal, -          ;                              [09]
                                097F  1423                  st_hexadecimal, -               ;                              [09]
                                097F  1424                  st_string, 10$, 10$, -          ;                              [09]
                                097F  1425                  10$, 10$, 10$, -                ;                              [09]
                                097F  1426                  st_logical>                     ;                              [09]
                    FF57   31   0998  1427 10$:    brw     extract                         ; Nop, just return            [09]
                                099B  1428 ST_DECIMAL:
                 53   56   D0  099B  1429          MOVL    R6,R3                           ; Prompt address
                      0082   30  099E  1430          bsbw    ex_common                       ; Do common                   [09]
      20 2E 4C 55 21 3D 43 41 21 00' 09A1  1431          .ASCIC  "!AC=!UL. "
                          09   09A1
                                09AB  1432 ST_OCTAL:
                 53   56   D0  09AB  1433          MOVL    R6,R3                           ; Prompt address
                        73   10  09AE  1434          BSBB    EX_COMMON                       ; Octal
29 4F 28 4C 4F 36 21 3D 43 41 21 00' 09B0  1435          .ASCIC  "!AC=!6OL(0) "
                          20   09BC
                          0C   09B0
                                09BD  1436 ST_HEXADECIMAL:
                 53   56   D0  09BD  1437          MOVL    R6,R3                           ; Prompt address
                        61   10  09C0  1438          BSBB    EX_COMMON                       ; Hexadecimal
20 29 58 28 4C 58 21 3D 43 41 21 00' 09C2  1439          .ASCIC  "!AC=!XL(X) "
                          0B   09C2
                                09CE  1440 ST_STRING:
                 53   56   D0  09CE  1441          MOVL    R6,R3                           ; Save address of prompt
                 50   86   9A  09D1  1442          MOVZBL  (R6)+,R0                        ; Get length of prompt
                 56   50   C0  09D4  1443 10$:     ADDL    R0,R6                           ; Skip previous string
                 50   86   9A  09D7  1444          MOVZBL  (R6)+,R0                        ; Get length of string
                      12   13  09DA  1445          BEQL    30$                             ; Branch if end
                   F5 59   F4  09DC  1446          SOBGEQ  R9,10$                          ; Count and branch
              59   FF A6   9E  09DF  1447          MOVAB   -1(R6),R9                       ; Address of prompt
                        3E   10  09E3  1448          BSBB    EX_COMMON                       ; Join common
20 43 41 21 3D 43 41 21 00' 09E5  1449          .ASCIC  "!AC=!AC "
                          08   09E5
                        33   10  09EE  1450 30$:     BSBB    EX_COMMON                       ; Join common
20 3F 3F 3F 3D 43 41 21 00' 09F0  1451          .ASCIC  "!AC=??? "
                          08   09F0
                                09F9  1452
                                09F9  1453 st_logical:                                     ;                              [09]
                 53   56   D0  09F9  1454          movl    r6, r3                          ; Get last prompt address     [09]
      50   00000A1A'EF   9E  09FC  1455          movab   20$, r0                         ; 'yes' string                [09]
                    07 59   E8  0A03  1456          blbs    r9, 10$                         ; If stored value is 1 (yes), OK [09]
      50   00000A1F'EF   9E  0A06  1457          movab   30$, r0                         ; Else set string to 'no'     [09]
                 59   50   D0  0A0D  1458 10$:     movl    r0, r9                          ; Set R9 to string            [09]
                      11   10  0A10  1459          bsbb    ex_common                       ; (SP) gives format string    [09]
   43 41 21 3D 43 41 21 00' 0A12  1460          .ascic  "!AC=!AC"                       ;                              [09]
                          07   0A12
            20 73 65 59 00' 0A1A  1461 20$:     .ascic  "Yes "                          ; Yes response                [09]
                          04   0A1A
               20 6F 4E 00' 0A1F  1462 30$:     .ascic  "No "                           ; No response                 [09]
                          03   0A1F
                                0A23  1463
                                0A23  1464 EX_COMMON:
                    02   BA   0A23  1465          POPR    #^MR1                           ; Address of edit string
                 50   81   9A  0A25  1466          MOVZBL  (R1)+,R0                        ; Get length
                 0183 8F   BB  0A28  1467          PUSHR   #^M<R0,R1,R7,R8>                ; Push R0/R1, buffer len,address
                                0A2C  1468
```

```
                 0208 8F    BB   0A2C  1469          PUSHR   #^M<R3,R9>              ; Push field name, value
                   10 AE    7F   0A30  1470          PUSHAQ  16(SP)                  ; Address of output descriptor
                   14 AE    DF   0A33  1471          PUSHAL  20(SP)                  ; Store length back in descriptor
                   10 AE    7F   0A36  1472          PUSHAQ  16(SP)                  ; Address of edit string
00000000'EF          05    FB   0A39  1473          CALLS   #5,L^SYS$FAO            ; Edit string
                                0A40  1474
                      CF    BA   0A40  1475          POPR    #^M<R0,R1,R2,R3>        ; Clean stack, get length in R2
                   57   52    C2   0A42  1476          SUBL    R2,R7                   ; Lessen available length
                   58   52    C0   0A45  1477          ADDL    R2,R8                   ; And skip over for next time
                        56    D4   0A48  1478          CLRL    R6                      ; Don't do again
                      FEA5    31   0A4A  1479          BRW     EXTRACT                 ; Next!
```

```
                                0A4D  1481          .SBTTL  GET_LINE         Condition getline
                                0A4D  1482
                                0A4D  1483 ;++
                                0A4D  1484 ; FUNCTIONAL DESCRIPTION:
                                0A4D  1485 ;
                                0A4D  1486 ;       This routine conditionally prompts for input with the
                                0A4D  1487 ;       prompt string provided, if any input is left from the last time
                                0A4D  1488 ;       it is used before prompting is done.
                                0A4D  1489 ;
                                0A4D  1490 ; CALLING SEQUENCE:
                                0A4D  1491 ;
                                0A4D  1492 ;       BSBB    GET_LINE
                                0A4D  1493 ;
                                0A4D  1494 ; INPUT PARAMETERS:     NONE
                                CA4D  1495 ;
                                0A4D  1496 ; IMPLICIT INPUTS:
                                0A4D  1497 ;
                                0A4D  1498 ;       R3      Address of prompt string (ASCIC)
                                0A4D  1499 ;       R4      Length remaining in last input string
                                0A4D  1500 ;       R5      Address of remaining input string
                                0A4D  1501 ;       R6      Address of prompt string output descriptor
                                0A4D  1502 ;       R8      Address of input line descriptor
                                0A4D  1503 ;
                                0A4D  1504 ; OUTPUT PARAMETERS:     NONE
                                0A4D  1505 ;
                                0A4D  1506 ; IMPLICIT OUTPUTS:
                                0A4D  1507 ;
                                0A4D  1508 ;       R4      Length of input string
                                0A4D  1509 ;       R5      Address of input string
                                0A4D  1510 ;
                                0A4D  1511 ; SIDE EFFECTS:          NONE
                                0A4D  1512 ;
                                0A4D  1513 ; COMPLETION CODES:      NONE
                                0A4D  1514 ;--
```

```
                        0A4D  1516
                        0A4D  1517 GET_LINE:
            53    DD    0A4D  1518          PUSHL   R3                      ; Save register
 00000000'EF    16    0A4F  1519          Jsb     L^SCAN$SPACES           ; Skip over leading spaces              [18]
  02 A8    50    A0    0A55  1520          ADDW2   R0,2(R8)                ; Add to length used so far
            53  8ED0   0A59  1521          POPL    R3                      ; Restore register
     50     01    D0    0A5C  1522          MOVL    S^#SS$_NORMAL,R0        ; Assume it worked
            54    D5    0A5F  1523          TSTL    R4                      ; Any input left?
            11    12    0A61  1524          BNEQ    10$                     ; Exit if some input left
            63    9F    0A63  1525          PUSHAB  (R3)                    ; Prompt string
 000000DD'EF    7F    0A65  1526          PUSHAQ  L^T_PROMPT              ; Edit string (add " ?")                [10]
   75'AF    02    FB    0A6B  1527          CALLS   #2,B^FORMAT_PROMPT      ; Format it
 50   0114 8F    3C    0A6F  1528          MOVZWL  #SS$_INSFARG,R0         ; Not enough arguments
                  05    0A74  1529 10$:     RSB
```

```
                        0A75  1531          .SBTTL  FORMAT_PROMPT   Format and load prompt
                        0A75  1532
                        0A75  1533 ;++
                        0A75  1534 ; Functional description:
                        0A75  1535 ;       Call FAO with arguments and format string given, store resultant
                        0A75  1536 ;       string in prompt output string
                        0A75  1537 ;
                        0A75  1538 ; Calling sequence:
                        0A75  1539 ;       FORMAT_PROMPT (FORMAT, params)
                        0A75  1540 ;
                        0A75  1541 ; Input parameters:
                        0A75  1542 ;       format          address of format descriptor
                        0A75  1543 ;       params          list of arguments to FAO
                        0A75  1544 ;
                        0A75  1545 ; Implicit inputs:
                        0A75  1546 ;       R6              contains pointer to output descriptor
                        0A75  1547 ;
                        0A75  1548 ; Outputs:
                        0A75  1549 ;       none
                        0A75  1550 ;
                        0A75  1551 ; Implicit outputs:
                        0A75  1552 ;       prompt string modified
                        0A75  1553 ;
                        0A75  1554 ; Side effects:
                        0A75  1555 ;       none
                        0A75  1556 ;
                        0A75  1557 ; Return codes:
                        0A75  1558 ;       FAO return codes (success, string truncated, or bad directive)
                        0A75  1559 ;--
```

```
                                0A75  1561
                                0A75  1562          $OFFSET 4,POSITIVE,<-
                                0A75  1563                  <FPT$_FORMAT,4>,-
                                0A75  1564                  <FPT$_ARGS>>
                                0004        FPT$_FORMAT:
                                0008        FPT$_ARGS:
                                0A75  1565
                                0A75  1566  FORMAT_PROMPT:
                          0008  0A75  1567          .WORD   ^M<R3>
              53   7E     7E    0A77  1568          MOVAQ   -(SP),R3                ; Allocate output descriptor
        63    66   01     A3    0A7A  1569          SUBW3   #1,(R6),(R3)            ; Set length
  04 A3    01    04 A6    C1    0A7E  1570          ADDL3   4(R6),#1,4(R3)          ; Address to put text
                                0A84  1571          $FAOL_S @FPT$_FORMAT(AP),(R3),- ; Do formatting
                                0A84  1572                   (R3),FPT$_ARGS(AP)
        66    63   01     A1    0A95  1573          ADDW3   #1,(R3),(R6)            ; Length including length byte
          04 B6   63     90    0A99  1574          MOVB    (R3),@4(R6)             ; Store result length
                          04    0A9D  1575          RET
```

```
                     0A9E  1577          .SBTTL  DSV$SHOWDEVICE  Display device information
                     0A9E  1578
                     0A9E  1579 ;++
                     0A9E  1580 ; FUNCTIONAL DESCRIPTION:
                     0A9E  1581 ;
                     0A9E  1582 ;       This routine is called from CLI to display information about
                     0A9E  1583 ;       devices attached to the system.
                     0A9E  1584 ;
                     0A9E  1585 ; CALLING SEQUENCE:
                     0A9E  1586 ;
                     0A9E  1587 ;       BSBW    DSV$SHOWDEVICE
                     0A9E  1588 ;
                     0A9E  1589 ; INPUT PARAMETERS:
                     0A9E  1590 ;
                     CA9E  1591 ; IMPLICIT INPUTS:
                     0A9E  1592 ;
                     0A9E  1593 ;       CLI$Q_FILE(R2)  Has descriptor for rest of command line.
                     0A9F  1594 ;
                     0A9E  1595 ; OUTPUT PARAMETERS:    NONE
                     0A9E  1596 ;
                     0A9E  1597 ; IMPLICIT OUTPUTS:     NONE
                     0A9E  1598 ;
                     0A9E  1599 ; SIDE EFFECTS:         NONE
                     0A9E  1600 ;
                     0A9E  1601 ; COMPLETION CODES:     NONE
                     0A9E  1602 ;
                     0A9E  1603 ; REGISTER USAGE:
                     0A9E  1604 ;
                     0A9E  1605 ;       R0      Length of device name as typed
                     0A9E  1606 ;       R1      Address of device name as typed
                     0A9E  1607 ;       R2      Length of Device name in DPB
                     0A9E  1608 ;       R3      Address of device name in DPB
                     0A9E  1609 ;       R5      Index into DPBLIST
                     0A9E  1610 ;       R6      Address of link device for /Adapter
                     0A9E  1611 ;       R10     DPB address
                     0A9E  1612 ;--
```

```
                          0A9E  1614
                          0A9E  1615 DSV$SHOWDEVICEB::                                                    ;                                                  [07]
00000000'EF   01    88    0A9E  1616          BISB2   #1@FLG$V_BRIEF,COMM_FLAGS  ; Set brief flag          [07]
                          0AA5  1617 DSV$SHOWDEVICE::
      50   08 A2    7D    0AA5  1618          MOVQ    CLI$Q_FILE(R2),R0         ; Get descriptor of input
         087F 8F    BB    0AA9  1619          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R11>   ;                    [11]
                          0AAD  1620
55   00000000'EF   DE    0AAD  1621          MOVAL   DS$GA_PTABLE,R5           ; Base PTABLE list
              04AF  30    0AB4  1622          bsbw    loc$adapter               ; Find /Adapter value, if any  [11]
           39 50    E9    0AB7  1623          blbc    r0, 70$                   ; If not found, return         [11]
        54    65    D0    0ABA  1624 10$:     MOVL    (R5),R4                   ; Get count of PT entries      [11]
     5B    6544    D0    0ABD  1625 20$:     MOVL    (R5)[R4],R11              ; Get address of Associated PT
              2D    13    0AC1  1626          BEQL    60$                       ; Branch if none
              56    D5    0AC3  1627          tstl    r6                        ; Was /Adapter specified?      [11]
              06    19    0AC5  1628          blss    30$                       ; If not, continue             [11]
     20 AB    56    D1    0AC7  1629          cmpl    r6, bn$a_link(r11)        ; Compare with link of current device  [11]
              23    12    0ACB  1630          bneq    60$                       ; If not, on to next device    [11]
        50    6E    D0    0ACD  1631 30$:     MOVL    (S.    .0                 ; Get length of input          [11]
              18    13    0AD0  1632          BEQL    50$                       ; Branch if Null input..equiv to all
     51    04 AE    D0    0AD2  1633          MOVL    4(SP),R1                  ; Get address of input
        52    6B    7D    0AD6  1634          MOVQ    HP$Q_DEVICE(R11),R2       ; Get length,address of device name
              52    D7    0AD9  1635          DECL    R2                        ; Remove " "
              53    D6    0ADB  1636          INCL    R3                        ; skip " "
        52    50    D1    0ADD  1637          CMPL    R0,R2                     ; Compare name lengths
              0F    14    0AE0  1638          BGTR    60$                       ; Skip compare if input longer than name
        83    81    91    0AE2  1639 40$:     CMPB    (R1)+,(R3)+               ; Compare characters
              09    12    0AE5  1640          BNEQ    60$                       ; Branch if mismatch
           F8 50    F5    0AE7  1641          SOBGTR  R0,40$                    ; Count and continue
                          0AEA  1642
   00000E39'EF    16    0AEA  1643 50$:     Jsb     L^SHOW_DEV                ; Display this DPB             [18]
                          0AF0  1644
         CA 54    F5    0AF0  1645 60$:     SOBGTR  R4,20$                    ; Count and branch if more
                          0AF3  1646
      087F 8F    BA    0AF3  1647 70$:     POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R11>   ;                    [11]
00000000'EF   01    8A    0AF7  1648          BICB2   #1@FLG$V_BRIEF,COMM_FLAGS  ; Clear brief flag before exit  [07]
                    05    0AFE  1649          RSB
```

```
                     0AFF  1651          .SBTTL  DSV$SHOWSELECT  Display information selects
                     0AFF  1652
                     0AFF  1653 ;++
                     0AFF  1654 ; FUNCTIONAL DESCRIPTION:
                     0AFF  1655 ;
                     0AFF  1656 ;       This routine is called from CLI to display information
                     0AFF  1657 ;       about device selected for test
                     0AFF  1658 ;
                     0AFF  1659 ; CALLING SEQUENCE:
                     0AFF  1660 ;
                     0AFF  1661 ;       BSBW    DSV$SHOWSELECT
                     0AFF  1662 ;
                     0AFF  1663 ; INPUT PARAMETERS:     NONE
                     0AFF  1664 ;
                     0AFF  1665 ; IMPLICIT INPUTS:
                     0AFF  1666 ;
                     0AFF  1667 ;       CLI$Q_FILE(R2)  Descriptor of device to be shown
                     0AFF  1668 ;
                     0AFF  1669 ; OUTPUT PARAMETERS:    NONE
                     0AFF  1670 ;
                     0AFF  1671 ; IMPLICIT OUTPUTS:     NONE
                     0AFF  1672 ;
                     0AFF  1673 ; COMPLETION CODES:     NONE
                     0AFF  1674 ;
                     0AFF  1675 ; SIDE EFFECTS:         NONE
                     0AFF  1676 ;
                     0AFF  1677 ; REGISTER USAGE:
                     0AFF  1678 ;
                     0AFF  1679 ;       R0/R1    Scratch
                     0AFF  1680 ;
                     0AFF  1681 ;       R10      Address of DPB for device to be shown
                     0AFF  1682 ;--
```

```
                            0AFF
                            0AFF  DSV$SHOWSELECTB::                                                        [07]
    00000000'EF    01   88  0AFF        BISB2   #1@FLG$V_BRIEF,COMM_FLAGS        ; Set brief flag          [07]
                            0B06  DSV$SHOWSELECT::
          080C 8F       BB  0B06        PUSHR   #^M<R2,R3,R11>
                            0B0A
    53   00000000'EF    DE  0B0A        MOVAL   DS$GA_PTABLE,R3           ; Base PTABLE LIST
               52  63   D0  0B11        MOVL    (R3),R2                   ; Fetch count of PT entries
                            0B14  10$:
 0C 00000000'EF    52   E1  0B14        BBC     R2,DS$GA_SELECTS,20$      ; Branch if not selected
          5B     6342   D0  0B1C        MOVL    (R3)[R2],R11              ; Address of PT
                   06   13  0B20        BEQL    20$                       ; Branch if none
         00000E39'EF    16  0B22        Jsb     L^SHOW_DEV                ; Display device information for this  [18]
                            0B28  20$:
              E9   52   F5  0B28        SOBGTR  R2,10$                    ; Count and Branch if more
          080C 8F       BA  0B2B        POPR    #^M<R2,R3,R11>            ; Restore
    00000000'EF    01   8A  0B2F        BICB2   #1@FLG$V_BRIEF,COMM_FLAGS        ; Clear brief flag before exit  [07]
                        05  0B36        RSB
```

```
                                0B37  1703          .Subtitle       DSV$DEATTACH    Deattach device
                                0B37  1704
                                0B37  1705 ;++
                                0B37  1706 ; FUNCTIONAL DESCRIPTION:
                                0B37  1707 ;
                                0B37  1708 ;       This routine is called from CLI to reverse the ATTACH command
                                0B37  1709 ;       for a given device.  Since the attach data base is a tree
                                0B37  1710 ;       structure, all subsidiary devices are also implicitly
                                0B37  1711 ;       deattached.
                                0B37  1712 ;
                                0B37  1713 ; CALLING SEQUENCE:
                                0B37  1714 ;
                                0B37  1715 ;       BSBW    DSV$DEATTACH
                                0B37  1716 ;
                                0B37  1717 ; INPUT PARAMETERS:     NONE
                                0B37  1718 ;
                                0B37  1719 ; IMPLICIT INPUTS:
                                0B37  1720 ;
                                0B37  1721 ;       CLI$Q_FILE      descriptor of device to be added
                                0B37  1722 ;       Q_ADAPTER       descriptor of adapter device is attached to
                                0B37  1723 ;
                                0B37  1724 ; OUTPUT PARAMETERS:    NONE
                                0B37  1725 ;
                                0B37  1726 ; IMPLICIT OUTPUTS:     NONE
                                0B37  1727 ;
                                0B37  1728 ; SIDE EFFECTS:         NONE
                                0B37  1729 ;
                                0B37  1730 ; COMPLETION CODES:     N/A
                                0B37  1731 ;--
                                0B37  1732
                                0B37  1733 dsv$deattach::                                               ;                               [12]
        0067 8F     BB          0B37  1734          pushr   #^M<r0,r1,r2,r5,r6>                         ; Save registers                [12]
    00000000'EF     16          0B3B  1735          jsb     script$flush                                ; Flush out scripts if should   [12]
55  00000000'EF     DE          0B41  1736          moval   ds$ga_ptable, r5                            ; Cache address of table        [12]
            041B    30          0B48  1737          bsbw    loc$adapter                                 ; Find the specified adapter    [12]
          36 50     E9          0B4B  1738          blbc    r0, 20$                                     ; If not found, return          [12]
       50   08 A2   7D          0B4E  1739          movq    cli$q_file(r2), r0                          ; Get device to deattach        [12]
          52   56   D0          0B52  1740          movl    r6, r2                                      ; Copy link address             [12]
            0463    30          0B55  1741          bsbw    loc$ptable                                  ; Locate it                     [12]
              28    12          0B58  1742          bneq    10$                                         ; If found, then it's OK        [12]
       52   08 AE   D0          0B5A  1743          movl    8(sp), r2                                   ; Restore CLI table pointer     [12]
7E  00000002'EF     7D          0B5E  1744          movq    q_adapter, -(sp)                            ; Get adapter name              [12]
          6E   6E   3C          0B65  1745          movzwl  (sp), (sp)                                  ; Make sure high word is 0      [12]
       7E   08 A2   7D          0B68  1746          movq    cli$q_file(r2), -(sp)                       ; Get device name               [12]
          6E   6E   3C          0B6C  1747          movzwl  (sp), (sp)                                  ; And zero extend length        [12]
    00000138'EF     9F          0B6F  1748          pushab  L^%t_no_such_on_adapter                     ; No such device                [12]
              01    DD          0B75  1749          pushl   #ds$k_printf                                ; Use PRINTF                    [12]
              15    DD          0B77  1750          pushl   #ds$k_type_command_err                      ; Message is command error      [12]
00000000'GF   07    FB          0B79  1751          calls   #7, G^dsx$print                             ; Print it                      [12]
              02    11          0B80  1752          brb     20$                                         ; Return                        [12]
              05    10          0B82  1753 10$:     bsbb    30$                                         ; Do the stuff, recursively     [12]
        0067 8F     BA          0B84  1754 20$:     popr    #^M<r0,r1,r2,r5,r6>                         ; Restore the registers         [12]
                    05          0B88  1755          rsb
```

```
                        0B89  1757          .Subtitle       Deattach subtree
                        0B89  1758 ;+
                        0B89  1759 ; This routine deallocates the Ptable for a device, clears the select bit
                        0B89  1760 ; and Ptable vector pointer for it; and if there were devices attached to
                        0B89  1761 ; it, does the same for each of them.
                        0B89  1762 ;-
                     1F BB  0B89  1763 30$:    pushr   #^M<r0,r1,r2,r3,r4>             ; Save some registers           [12]
                        0B8B  1764 ;
                        0B8B  1765 ; See if anything attached to this one (R1)
                        0B8B  1766 ;
                  50   65 D0  0B8B  1767          movl    (r5), r0                        ; Size of table                [12]
                51   6540 D0  0B8E  1768 40$:    movl    (r5)[r0], r1                    ; Get address of next Ptable   [12]
                     09 13  0B92  1769          beql    50$                             ; Skip if gone                 [12]
             04 AE   20 A1 D1  0B94  1770          cmpl    hp$a_link(r1), 4(sp)            ; Check against input device   [12]
                     02 12  0B99  1771          bneq    50$                             ; If not linked to it, skip    [12]
                     EC 10  0B9B  1772          bsbb    30$                             ; Do same for IT, recursively  [12]
                  EE 50 F5  0B9D  1773 50$:    sobgtr  r0, 40$                 ; Loop through the table        [12]
                        0BA0  1774          $print  #ds$k_type_command_out, -       ; Print command info           [12]
                        0BA0  1775                    #ds$k_printf, -               ; .. use PRINTF                [12]
                        0BA0  1776                    L^t_deattach, -               ; .. Deattach message          [12]
                        0BA0  1777                    4(sp)                         ; .. address of Ptable         [12]
                  50   6E 7D  0BB6  1778          movq    (sp), r0                        ; Get original r0 and r1       [12]
                     6540 D4  0BB9  1779          clrl    (r5)[r0]                        ; Clear table pointer          [12]
          00 00000000'EF   50 E5  0BBC  1780          bbcc    r0, L^ds$ga_selects, 60$        ; Clear select bit for it      [12]
                  50   51 D0  0BC4  1781 60$:    movl    r1, r0                          ; Copy address of Ptable       [12]
               00000000'EF   16  0BC7  1782          Jsb     L^exe$deanonpaged               ; Deallocate it                [18]
                     1F BA  0BCD  1783          popr    #^M<r0,r1,r2,r3,r4>             ; Restore registers            [12]
                        05  0BCF  1784          rsb                                     ; Return                       [12]
```

```
                     0BD0  1786          .SBTTL  DSV$SELECT      Select device for testing
                     0BD0  1787
                     0BD0  1788 ;++
                     0BD0  1789 ; FUNCTIONAL DESCRIPTION:
                     0BD0  1790 ;
                     0BD0  1791 ;       This routine is called from DS$CLI for each device to be
                     0BD0  1792 ;       tested.  It sets the select bit and moves the P-table
                     0BD0  1793 ;       entry to the end of the P-table list
                     0BD0  1794 ;
                     0BD0  1795 ; CALLING SEQUENCE:
                     0BD0  1796 ;
                     0BD0  1797 ;       BSBW    DSV$SELECT
                     0BD0  1798 ;
                     0BD0  1799 ; INPUT PARAMETERS:     NONE
                     0BD0  1800 ;
                     0BD0  1801 ; IMPLICIT INPUTS:
                     0BD0  1802 ;
                     0BD0  1803 ;       CLI$Q_FILE      descriptor of device to be added
                     0BD0  1804 ;
                     0BD0  1805 ; OUTPUT PARAMETERS:    NONE
                     0BD0  1806 ;
                     0BD0  1807 ; IMPLICIT OUTPUTS:
                     0BD0  1808 ;
                     0BD0  1809 ;       DS$GA_PTABLE    The indicated unit is moved to the head.
                     0BD0  1810 ;
                     0BD0  1811 ; SIDE EFFECTS:         NONE
                     0BD0  1812 ;
                     0BD0  1813 ; COMPLETION CODES:     N/A
                     0BD0  1814 ;--
                     0BD0  1815
                     0BD0  1816 DSV$SELECT::
      007F 8F    BB  0BD0  1817          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6>       ; Preserve                              [11]
 00000000'EF     16  0BD4  1818          JSB     SCRIPT$FLUSH                    ; Flush scripts if console command      [10]
        0302     30  0BDA  1819          BsbW    CheckAmbiguous                  ; Check for ambiguous names             [15]
       14 50     E9  0BDD  1820          BlbC    R0, 10$                         ; If error, hopscotch on to exit        [15]
55 00000000'EF     DE  0BE0  1821          MOVAL   DS$GA_PTABLE,R5                 ; Point to P-table
53 00000000'EF     DE  0BE7  1822          MOVAL   DS$GA_SELECTS,R3                ; Point to select bit mask
        0375     30  0BEE  1823          bsbw    loc$adapter                     ; Find /Adapter, if any                 [11]
       03 50     E8  0BF1  1824          blbs    r0, 20$                         ; Continue if OK                        [11]
        00DB     31  0BF4  1825 10$:     brw     140$                            ; Return if not                         [11]
   50   08 A2     7D  0BF7  1826 20$:     MOVQ    CLI$Q_FILE(R2),R0               ; Descriptor of device                  [11]
          44     13  0BFB  1827          BEQL    60$                             ; Exit if no device
52   61   08     9C  0BFD  1828          ROTL    #8,(R1),R2                      ; Get first 3 characters of name
     52   50     90  0C01  1829          MOVB    R0,R2                           ; Insert length
52 4C4C4103 8F     D1  0C04  1830          CMPL    #<^A"@ALL"-61>,R2               ; compare with ascic "all"
          37     12  0C0B  1831          BNEQ    70$                             ; Branch if not all
```

```
                     0C0D  1833
                     0C0D  1834 ;+
                     0C0D  1835 ; Process SELECT ALL command
                     0C0D  1836 ;-
      54    65   D0  0C0D  1837          MOVL    (R5),R4                 ; Get count of devices possible
   50    6544   D0  0C10  1838 30$:      MOVL    (R5)[R4],R0             ; Anything here?
            28   13  0C14  1839          BEQL    50$                     ; No check next
            56   D5  0C16  1840          tstl    r6                      ; If no /Adapter                        [11]
            06   19  0C18  1841          blss    40$                     ; ...then branch around the check       [11]
   56    20 A0   D1  0C1A  1842          cmpl    hp$a_link(r0), r6       ; Is this on the right link?            [11]
            1E   12  0C1E  1843          bneq    50$                     ; If not, loop to next                  [11]
      50    60   7D  0C20  1844 40$:      MOVQ    HP$Q_DEVICE(R0),R0      ; Get device name                       [11]
            50   D7  0C23  1845          DECL    R0                      ; remove "-" from length
            51   D6  0C25  1846          INCL    R1                      ; remove "-" from address
00000000'EF      16  0C27  1847          Jsb     L^DSR$IFLOADDEV         ; Is this the load device?              [18]
      0E 50      E8  0C2D  1848          BLBS    R0,50$                  ; Branch if it is
   0A 63   54    E2  0C30  1849          BBSS    R4,(R3),50$             ; Branch if already set and set
         00A0    30  0C34  1850          BSBW    150$                    ; Allocate the device if necessary      [11]
      04 50      E8  0C37  1851          BLBS    R0,50$                  ; Branch if successful
   00 63   54    E5  0C3A  1852          BBCC    R4,(R3),50$             ; clear select bit if allocate failed
         CF 54    F5  0C3E  1853 50$:     SOBGTR  R4,30$                  ; Do them all
          008E    31  0C41  1854 60$:     BRW     140$                    ; Branch to exit                        [11]
                     0C44  1855
```

```
                                0C44  1857
                                0C44  1858 ;+
                                0C44  1859 ; Process SELECT device command
                                0C44  1860 ;-
            52    01    CE    0C44  1861 70$:     mnegl   #1, r2                      ; Set wildcard link                         [09]
                  56    D5    0C47  1862          tstl    r6                          ; If there's no /Adapter,                   [11]
                  03    19    0C49  1863          blss    80$                         ; ...then don't check it                    [11]
            52    56    D0    0C4B  1864          movl    r6, r2                      ; If is, use address of link device         [11]
               036A     30    0C4E  1865 80$:     bsbw    loc$ptable                  ; Search for it                             [11]
                  46    12    0C51  1866          BNEQ    110$                        ; Branch if found
         52    08 AE    D0    0C53  1867          MOVL    8(SP),R2                    ; Get base of CLI
                  56    D5    0C57  1868          tstl    r6                          ; /ADAPTER?                                 [11]
                  24    19    0C59  1869          blss    90$                         ; Branch if not                             [11]
      7E  00000002'EF    7D    0C5B  1870          movq    g_adapter, -(sp)            ; Push on adapter name                      [11]
            6E    6E    3C    0C62  1871          movzwl  (sp), (sp)                  ; Zero extend                               [11]
         7E    08 A2    7D    0C65  1872          movq    cli$q_file(r2), -(sp)       ; Push on device name                       [11]
            6E    6E    3C    0C69  1873          movzwl  (sp), (sp)                  ; Zero extend                               [11]
         00000138'EF    9F    0C6C  1874          pushab  L^t_no_such_on_adapter      ; Format string                             [11]
                  01    DD    0C72  1875          pushl   #ds$k_printf                ; Use printf                                [13]
                  15    DD    0C74  1876          pushl   #ds$k_type_command_err      ; Command error type code                   [13]
   00000000'GF    07    FB    0C76  1877          calls   #7, G^dsx$print             ; Print it                                  [13]
                  53    11    0C7D  1878          brb     140$                        ; Exit                                      [11]
         7E    08 A2    7D    0C7F  1879 90$:     MOVQ    CLI$Q_FILE(R2),-(SP)        ; Stack file name                           [11]
            6E    6E    3C    0C83  1880 100$:    movzwl  (sp), (sp)                  ; Zero extend length                        [11]
         0000010E'EF    9F    0C86  1881          PUSHAB  L^T_NO_SUCH_DEV             ; Edit string                               [11]
                  01    DD    0C8C  1882          pushl   #ds$k_printf                ; Use PRINTF                                [13]
                  15    DD    0C8E  1883          pushl   #ds$k_type_command_err      ; Command error type code                   [13]
   00000000'GF    05    FB    0C90  1884          CALLS   #5, G^DSX$PRINT             ; Type the error                            [13]
                  39    11    0C97  1885          BRB     140$                        ; Branch to exit
                                0C99  1886 110$:
```

```
                                0C99  1888
                                0C99  1889 ;+
                                0C99  1890 ; Compress P-table list to high end of table
                                0C99  1891 ; R1 = P-table address
                                0C99  1892 ;-
                    54    50    D0  0C99  1893          MOVL    R0,R4                   ; Copy unit index of device
                          39    10  0C9C  1894          BSBB    150$                    ; Allocate device if necessary      [11]
                       31 50    E9  0C9E  1895          BLBC    R0,140$                 ; Branch if can't allocate
                        6544    DD  0CA1  1896          PUSHL   (R5)[R4]                ; Save address of P-table
                        6544    D4  0CA4  1897          CLRL    (R5)[R4]                ; Clear address of P-table
                    54    65    D0  0CA7  1898          MOVL    (R5),R4                 ; Get count of entries
                    52    54    D0  0CAA  1899          MOVL    R4,R2                   ; current position
                                0CAD  1900
                50    6542    D0  0CAD  1901 120$:    MOVL    (R5)[R2],R0             ; Copy address to first available
                          13    13  0CB1  1902          BEQL    130$                    ; Empty, next
                        6542    D4  0CB3  1903          CLRL    (R5)[R2]                ; Clear it out
                    6544    50    D0  0CB6  1904          MOVL    R0,(R5)[R4]             ; Put it in the first available slot
          50    63    01    52    EF  0CBA  1905          EXTZV   R2,#1,(R3),R0           ; Get select bit
          63    01    54    50    F0  0CBF  1906          INSV    R0,R4,#1,(R3)           ; Insert new select bit
                          54    D7  0CC4  1907          DECL    R4                      ; Use previous entry next time
                       E4 52    F5  0CC6  1908 130$:    SOBGTR  R2,120$                 ; Decrement and continue
                                0CC9  1909
                        6544 8ED0  0CC9  1910          POPL    (R5)[R4]                ; Insert unit at available slot
          63    01    54    01    F0  0CCD  1911          INSV    #1,R4,#1,(R3)           ; Set the select bit
                                0CD2  1912 140$:
                    007F 8F    BA  0CD2  1913          POPR    #^M<R0,R1,R2,R3,R4,R5,R6>       ; Restore                   [11]
                               05  0CD6  1914          RSB
                                0CD7  1915
```

```
                                0CD7  1917
                                0CD7  1918 ;+
                                0CD7  1919 ; Allocate device (R5)[R4] if usermode and should be allocated
                                0CD7  1920 ; and not already allocated.  Produce error messages if it fails.
                                0CD7  1921 ;-
            50    01    D0    0CD7  1922 150$:   MOVL    S^#SS$_NORMAL,R0        ; Assume success. esp. if S/A
0000FE00'EF       1C    E1    0CDA  1923         BBC     #DSA$V_USER, -
                  73          0CE1  1924                 DSA$GL_FLAGS, 180$      ; Exit if not usermode
                              0CE2  1925         Br_If_Inhibit_Allocate  180$   ; If bit set in DS$GL_CRD_Flags,
                              0CEA  1926                                         ; inhibit device allocation          [22]
        52    6544    D0    0CEA  1927         MOVL    (R5)[R4],R2             ; Get address of P-table
   62 0A A2     00    E1    0CEE  1928         BBC     #HP$V_ALLOC, -
                              0CF3  1929                 HP$B_FLAGS(R2), 180$    ; Branch if should not be allocated
   5D 0A A2     01    E2    0CF3  1930         BBSS    #HP$V_WASALL, -
                              0CF8  1931                 HP$B_FLAGS(R2), 180$    ; Branch if already allocated
                              0CF8  1932         $ALLOC_S HP$Q_DEVICE(R2)        ; Address of device name
    50   0641  8F    B1    0D07  1933         CMPW    #SS$_DEVALRALLOC,R0     ; Already allocated to us?
                  05    12    0D0C  1934         BNEQ    160$                    ; Branch if not
   42 0A A2     01    E4    0D0E  1935         BBSC    #HP$V_WASALL, -
                              0D13  1936                 HP$B_FLAGS(R2), 180$    ; Clear allocated bit and exit
            3F 50    E8    0D13  1937 160$:   BLBS    R0,180$                 ; Branch if successful
      0A A2     02    8A    0D16  1938         BICB    #HP$M_WASALL, -
                              0D1A  1939                 HP$B_FLAGS(R2)          ; Clear allocated bit
                  50    DD    0D1A  1940         PUSHL   R0                      ; Save return code
              7E    62    7D    0D1C  1941         MOVQ    HP$Q_DEVICE(R2),-(SP)   ; device name arg for print
    0000010E'EF       9F    0D1F  1942         PUSHAB  L^T_NO_SUCH_DEV         ; Assume no such device              [10]
    50   0908  8F    B1    0D25  1943         CMPW    #SS$_NOSUCHDEV,R0       ; No such device?
                  1B    13    0D2A  1944         BEQL    170$                    ; Branch if truth
6E  000001E1'EF       9E    0D2C  1945         MOVAB   L^T_DEVALLOC,(SP)       ; Assume allocated to other user     [10]
    50   0840  8F    B1    0D33  1946         CMPW    #SS$_DEVALLOC,R0        ; Already allocated to another?
                  0D    13    0D38  1947         BEQL    170$                    ; Branch if truth
              5E    0C    C0    0D3A  1948         ADDL    #12,SP                  ; Remove descriptor of name and text
                  50 8ED0   0D3D  1949         POPL    R0                      ; Restore completion code
     00000000'EF      16    0D40  1950         Jsb     L^DSR$COMPLETION        ; Other errors                       [18]
                  05    0D46  1951         RSB                             ; Return
                        0D47  1952
                  01    DD    0D47  1953 170$:   pushl   #ds$k_printf            ; Use printf to print                [13]
                  15    DD    0D49  1954         pushl   #ds$k_type_command_err  ; Command error type code            [13]
00000000'GF       05    FB    0D4B  1955         CALLS   #5, G^DSX$PRINT         ; Print the error                    [13]
                  50 8ED0   0D52  1956         POPL    R0                      ; Restore completion code
                  05    0D55  1957 180$:   RSB
```

```
                                0D56  1959          .SBTTL  DSV$DESELECT    Deselect device for testing
                                0D56  1960
                                0D56  1961 ;++
                                0D56  1962 ; FUNCTIONAL DESCRIPTION:
                                0D56  1963 ;
                                0D56  1964 ;       This routine is called from DS$CLI for each device to be
                                0D56  1965 ;       remved from testing.  It clears the select bit
                                0D56  1966 ;
                                0D56  1967 ; CALLING SEQUENCE:
                                0D56  1968 ;
                                0D56  1969 ;       BSBW    DSV$DESELECT
                                0D56  1970 ;
                                0D56  1971 ; INPUT PARAMETERS:     NONE
                                0D56  1972 ;
                                0D56  1973 ; IMPLICIT INPUTS:
                                0D56  1974 ;
                                0D56  1975 ;       CLI$Q_FILE      descriptor of device to be removed
                                0D56  1976 ;
                                0D56  1977 ; OUTPUT PARAMETERS:    NONE
                                0D56  1978 ;
                                0D56  1979 ; IMPLICIT OUTPUTS:     NONE
                                0D56  1980 ;
                                0D56  1981 ; SIDE EFFECTS:         NONE
                                0D56  1982 ;
                                0D56  1983 ; COMPLETION CODES:     N/A
                                0D56  1984 ;
                                0D56  1985 ; REGISTER USAGE:
                                0D56  1986 ;--
```

```
                                 0D56  1988
                                 0D56  1989 DSV$DESELECT::
                  007F 8F    BB  0D56  1990          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6>       ; Preserve                                  [11]
              00000000'EF    16  0D5A  1991          JSB     SCRIPT$FLUSH                    ; Flush scripts if console command          [10]
                     017C    30  0D60  1992          BsbW    CheckAmbiguous                  ; Check for ambiguous names                 [15]
                   14 50     E9  0D63  1993          BlbC    R0, 10$                         ; If error, hopscotch on to exit            [15]
          55   00000000'EF    DE  0D66  1994          MOVAL   DS$GA_PTABLE,R5                 ; Point to P-table
          53   00000000'EF    DE  0D6D  1995          MOVAL   DS$GA_SELECTS,R3                ; Point to select bit mask
                     01EF    30  0D74  1996          bsbw    loc$adapter                     ; Find /Adapter, if any                     [11]
                   03 50     E8  0D77  1997          blbs    r0, 20$                         ; Continue if OK                            [11]
                     0095    31  0D7A  1998 10$:     brw     120$                            ; Return elsewise                           [11]
                50   08 A2    7D  0D7D  1999 20$:     MOVQ    CLI$Q_FILE(R2),R0               ; Descriptor of device                      [11]
                        2F    13  0D81  2000          BEQL    60$                             ; Branch if no device                       [11]
             52   61    08    9C  0D83  2001          ROTL    #8,(R1),R2                      ; Get first 3 characters of name
                  52    50    90  0D87  2002          MOVB    R0,R2                           ; Insert length
          52   4C4C4103 8F    D1  0D8A  2003          CMPL    #<^A"@ALL"-61>,R2               ; compare with ascic "all"
                        21    12  0D91  2004          BNEQ    70$                             ; Branch if not all
```

```
                        0D93  2006
                        0D93  2007 ;+
                        0D93  2008 ; Process DESELECT ALL command
                        0D93  2009 ;-
          54   65   D0  0D93  2010          MOVL    (R5),R4                 ; Get count of devices possible
             6544   D5  0D96  2011 30$:     TSTL    (R5)[R4]                ; Anything here?
               14   13  0D99  2012          BEQL    50$                     ; No check next
               56   D5  0D9B  2013          tstl    r6                      ; If no /Adapter,                       [11]
               0A   19  0D9D  2014          blss    40$                     ; ...don't check it                     [11]
       50   6544   D0  0D9F  2015          movl    (r5)[r4], r0            ; Get address of link Ptable            [11]
    56   20 A0   D1  0DA3  2016          cmpl    hp$a_link(r0), r6       ; Compare                               [11]
               06   12  0DA7  2017          bneq    50$                     ; If not equal, loop to next            [11]
    02 63   54   E5  0DA9  2018 40$:     BBCC    R4,(R3),50$             ; Branch if already clear and clear     [11]
               68   10  0DAD  2019          BSBB    130$                    ; Deallocate the device if necessary
            E4 54   F5  0DAF  2020 50$:     SOBGTR  R4,30$                  ; Do them all
               5E   11  0DB2  2021 60$:     BRB     120$                    ; Branch to exit                        [11]
                        0DB4  2022
```

```
                                 0DB4  2024
                                 0DB4  2025 ;+
                                 0DB4  2026 ; Process DESELECT device command
                                 0DB4  2027 ;-
            52   01    CE       0DB4  2028 70$:     mnegl   #1, r2                          ; Set wildcard link                    [09]
                 56    D5       0DB7  2029          tstl    r6                              ; If no /Adapter,                      [11]
                 03    19       0DB9  2030          blss    80$                             ; ...don't check for it                [11]
            52   56    D0       0DBB  2031          movl    r6, r2                          ; Use /ADAPTER address                 [11]
               01FA    30       0DBE  2032 80$:     bsbw    loc$ptable                      ; Search for it                        [09]
                 46    12       0DC1  2033          BNEQ    110$                            ; Branch if found
         52    08 AE   D0       0DC3  2034          MOVL    8(SP),R2                        ; Get base of CLI
                 56    D5       0DC7  2035          tstl    r6                              ; /ADAPTER?                            [11]
                 24    19       0DC9  2036          blss    90$                             ; Branch if not                        [11]
      7E  00000002'EF  7D       0DCB  2037          movq    g_adapter, -(sp)                ; Push on adapter name                 [11]
            6E   6E    3C       0DD2  2038          movzwl  (sp), (sp)                      ; Zero extend                          [11]
         7E   08 A2    7D       0DD5  2039          movq    cli$q_file(r2), -(sp)           ; Push on device name                  [11]
            6E   6E    3C       0DD9  2040          movzwl  (sp), (sp)                      ; Zero extend                          [11]
         0000013B'EF   9F       0DDC  2041          pushab  L^t_no_such_on_adapter          ; Format string                        [11]
                 01    DD       0DE2  2042          pushl   #ds$k_printf                    ; Use PRINTF                           [13]
                 15    DD       0DE4  2043          pushl   #ds$k_type_command_err          ; Command error type code              [13]
   00000000'GF   07    FB       0DE6  2044          calls   #7, G^dsx$print                 ; Print it                             [13]
                 23    11       0DED  2045          brb     120$                            ; Exit                                 [11]
         7E   08 A2    7D       0DEF  2046 90$:     MOVQ    CLI$Q_FILE(R2),-(SP)            ; Stack file name                      [11]
            6E   6E    3C       0DF3  2047 100$:    movzwl  (sp), (sp)                      ; Zero extend                          [11]
         0000010E'EF   9F       0DF6  2048          PUSHAB  L^T_NO_SUCH_DEV                 ; Edit string                          [11]
                 01    DD       0DFC  2049          pushl   #ds$k_printf                    ; Use PRINTF                           [13]
                 15    DD       0DFE  2050          pushl   #ds$k_type_command_err          ; Command error type code              [13]
   00000000'GF   05    FB       0E00  2051          CALLS   #5, G^DSX$PRINT                 ; Type the error                       [13]
                 09    11       0E07  2052          BRB     120$                            ; Branch to exit
                                 0E09  2053 110$:
            54   50    D0       0E09  2054          MOVL    R0,R4                           ; Copy index
         02 63   54    E5       0E0C  2055          BBCC    R4,(R3),120$                    ; Branch if not selected
                 05    10       0E10  2056          bsbb    130$                            ; Deallocate the device if necessary   [11]
               007F    8F   BA  0E12  2057 120$:    POPR    #^M<R0,R1,R2,R3,R4,R5,R6>               ; Restore                      [11]
                 05    0E16  2058          RSB                                             ; Return
                       0E17  2059
```

```
                                 0E17  2061
                                 0E17  2062 ;+
                                 0E17  2063 ; Deallocate the device (R5)[R4] if usermode and should be deallocated
                                 0E17  2064 ; and was allocated.
                                 0E17  2065 ;-
0000FE00'EF    1C    E1          0E17  2066 130$:   BBC      #DSA$V_USER, -
                   19            0E1E  2067                  DSA$GL_FLAGS, 140$      ; Exit if not usermode
          50   6544   D0          0E1F  2068          MOVL    (R5)[R4],R0             ; Get address of P-table
      10 0A A0   00    E1          0E23  2069          BBC     #HP$V_ALLOC, -
                                   0E28  2070                  HP$B_FLAGS(R0), 140$    ; Branch if should not be allocated
      0B 0A A0   01    E5          0E28  2071          BBCC    #HP$V_WASALL, -
                                   0E2D  2072                  HP$B_FLAGS(R0), 140$    ; Branch if already allocated
                                   0E2D  2073          $DALLOC_S HP$Q_DEVICE(R0)       ; Deallocate the device
                         05        0E38  2074 140$:   RSB                             ; Return
```

```
                        0E39  2077          .SUBTITLE       SHOW_DEV Routine
                        0E39  2078
```

```
                     ;++
                     ; FUNCTIONAL DESCRIPTION:
                     ;
                     ;       This routine types out a line describing a single device.
                     ;
                     ; CALLING SEQUENCE:
                     ;
                     ;       BSBW    SHOW_DEV
                     ;
                     ; INPUT PARAMETERS:      NONE
                     ;
                     ; IMPLICIT INPUTS:
                     ;
                     ;       R10     Address of DPB for device
                     ;               Contents of DS$GA_TYPLIST
                     ;
                     ; OUTPUT PARAMETERS:     NONE
                     ;
                     ; IMPLICIT OUTPUTS:      NONE
                     ;
                     ; SIDE EFFECTS:          NONE
                     ;
                     ; REGISTER USAGE:
                     ;
                     ;       R11     Address of P-table for device to be displayed
                     ;--
```

```
                                0E39  2107
                     00000100   0E39  2108 SHOWBUF_SIZ     = 256                      ; Length of buffer
                                0E39  2109 SHOW_DEV:
              5E    FF00 CE     9E  0E39  2110             MOVAB   -SHOWBUF_SIZ(SP),SP ; Allocate buffer space
                         6E     9F  0E3E  2111             PUSHAB  (SP)                ; Address of buffer
              7E    0100 8F     3C  0E40  2112             MOVZWL  #SHOWBUF_SIZ,-(SP)  ; Length of buffer
                                0E45  2113
                51    26 AB     9E  0E45  2114             MOVAB   HP$T_TYPE(R11),R1   ; Get address of ascic type string
                      50  61     9A  0E49  2115             MOVZBL  (R1)+,R0            ; Get length
                0000100A'EF      16  0E4C  2116             Jsb     L^LOC$PTDESC        ; Locate the descriptor                 [18]
                         0D      12  0E52  2117             BNEQ    10$                 ; Branch if located
                                0E54  2118
        08 AE    2E2E2E2E  8F     D0  0E54  2119             MOVL    #^A"....",8(SP)     ; String is "...."
                      6E     03  D0  0E5C  2120             MOVL    #3,(SP)             ; Set length of string
                         12     11  0E5F  2121             BRB     20$                 ; Branch to print
                                0E61  2122
                         6E     7F  0E61  2123 10$:        PUSHAQ  (SP)                ; Address of descriptor
                         5B     DD  0E63  2124             PUSHL   R11                 ; Address of P-table
                      51    80     9A  0E65  2125             MOVZBL  (R0)+,R1            ; Get length of name
                      04 A041     9F  0E68  2126             PUSHAB  4(R0)[R1]           ; Push address of start code
           FFFFFA6D   EF     03     FB  0E6C  2127             CALLS   #3,L^PT$EXTRACT     ; Convert to ascii                      [10]
                                0E73  2128
        2F 00000000'EF     00     E0  0E73  2129 20$:        BBS     #FLG$V_BRIEF,COMM_FLAGS,50$ ; Branch if brief form.    [07]
                         6E     7F  0E7B  2130             PUSHAQ  (SP)                ; Address of descriptor                 [07]
                   18 AB     DD  0E7D  2131             PUSHL   HP$A_DEVICE(R11)    ; Stuff device address
                   20 BB     7F  0E80  2132             PUSHAQ  @HP$A_LINK(R11)     ; Address of link name quad desc
                         07     12  0E83  2133             BNEQ    30$                 ; Branch if descriptor
           6E    0000029E'EF      7E  0E85  2134             MOVAQ   Q_HUB,(SP)          ; Use HUB device name                   [11]
                   26 AB     9F  0E8C  2135 30$:        PUSHAB  HP$T_TYPE(R11)      ; Address of ascic type
                         6B     7F  0E8F  2136             PUSHAQ  HP$Q_DEVICE(R11)    ; Address of ascid device name
                00000202'EF      9F  0E91  2137             PUSHAB  L^T_$SHOWDEV        ; Address of edit string                [10]
                         01     DD  0E97  2138             pushl   #ds$k_printf        ; Use PRINTF                            [13]
                         16     DD  0E99  2139             pushl   #ds$k_type_command_out ; Command output type code          [13]
        00000000'GF      08     FB  0E9B  2140             CALLS   #8, G^DSX$PRINT     ; Print the line                        [13]
                                0EA2  2141
           5E    00000108 8F     C0  0EA2  2142 40$:        ADDL    #SHOWBUF_SIZ+8,SP   ; Remove descriptor and buffer        [07]
                         05     0EA9  2143             RSB                         ; Return
                   26 AB     9F  0EAA  2144 50$:        PUSHAB  HP$T_TYPE(R11)      ; Address of ascic type                 [07]
                         6B     7F  0EAD  2145             PUSHAQ  HP$Q_DEVICE(R11)    ; Address of ascid device name          [07]
                0000021C'EF      9F  0EAF  2146             PUSHAB  L^T_$SHOWDEVB       ; Address of edit string                [07]
                         01     DD  0EB5  2147             pushl   #ds$k_printf        ; Use PRINTF                            [13]
                         16     DD  0EB7  2148             pushl   #ds$k_type_command_out ; Command output type code          [13]
        00000000'GF      05     FB  0EB9  2149             CALLS   #5, G^DSX$PRINT     ; Print the line                        [13]
                         E0     11  0EC0  2150             BRB     40$                 ;                                       [07]
```

```
                        0EC2  2152          .SBTTL  INI$PTABLE      Initialize P-table
                        0EC2  2153
                        0EC2  2154 ;++
                        0EC2  2155 ; FUNCTIONAL DESCRIPTION:
                        0EC2  2156 ;
                        0EC2  2157 ;       This routine is used to initialize DS$GA_PTABLE and
                        0EC2  2158 ;       DS$GA_SELECTS, DS$GA_SELECTED.
                        0EC2  2159 ;
                        0EC2  2160 ; CALLING SEQUENCE:
                        0EC2  2161 ;
                        0EC2  2162 ;       BSBW    INI$PTABLE
                        0EC2  2163 ;
                        0EC2  2164 ; INPUT PARAMETERS:     NONE
                        0EC2  2165 ;
                        0EC2  2166 ; IMPLICIT INPUTS:      NONE
                        0EC2  2167 ;
                        0EC2  2168 ; OUTPUT PARAMETERS:    NONE
                        0EC2  2169 ;
                        0EC2  2170 ; IMPLICIT OUTPUTS:
                        0EC2  2171 ;
                        0EC2  2172 ;       DS$GA_SELECTS, DS$GA_SELECTED, DS$GA_PTABLE.
                        0EC2  2173 ;
                        0EC2  2174 ; SIDE EFFECTS:         NONE
                        0EC2  2175 ;
                        0EC2  2176 ; COMPLETION CODES:     SS$_NORMAL
                        0EC2  2177 ;--
```

```
                                0EC2  2179
                                0EC2  2180 INI$PTABLE::
            51  00000000'EF  DE   0EC2  2181          MOVAL   DS$GA_PTABLE,R1          ; Point to table
                      50  61  D0   0EC9  2182          MOVL    (R1),R0                 ; Get count
                        6140  D4   0ECC  2183 10$:      CLRL    (R1)[R0]
00000000'EF  01  50  00  F0   0ECF  2184          INSV    #0,R0,#1,DS$GA_SELECTS  ; Clear select bit
                        F1 50  F5   0ED8  2185          SOBGTR  R0,10$                  ; Do next unit
                      50  01  D0   0EDB  2186          MOVL    S^#SS$_NORMAL,R0        ; Success of course
                             05   0EDE  2187          RSB
                                0EDF  2188
```

```
                        .Subtitle       Check ambiguous device

;++
; Functional Description:
;
;       This routines searches to make sure that the /Adapter
;       argument (if specified) is unique; and that if /Adapter
;       is NOT used, the device name is unique.
;
; Calling Sequence:
;
;       BsbW    CheckAmbiguous
;
; Input Parameters:     NONE
;
; Implicit Inputs:      NONE
;
; Output Parameters:    NONE
;
; Implicit Outputs:     NONE
;
; Status Return:        R0 (LBS if OK, LBC if error)
;
;--
```

```
                        0EDF  2215
                        0EDF  2216 CheckAmbiguous:                              ; Check for ambiguous names                [15]
           007F 8F   BB 0EDF  2217          PushR   #^M<r0,r1,r2,r3,r4,r5,R6>   ;       ; Save registers                    [15]
                56   D4 0EE3  2218          ClrL    R6                         ; Assume we'll skip ahead to device        [15]
        0B 62   18   E1 0EE5  2219          Bbc     #Cli$V_Adapter, -           ; If no /Adapter, don't have               [15]
                        0EE9  2220                   Cli$L_Flags(R2), 10$       ; .. to check adapter name                 [15]
                56   D6 0EE9  2221          IncL    R6                         ; Have to check adapter first              [15]
6E   00000002'EF   7D 0EEB  2222          MovQ    Q_Adapter, (Sp)            ; Set save descriptor to adapter name      [15]
                0C   11 0EF2  2223          Brb     20$                        ; Join common                              [15]
        52   08 AE   D0 0EF4  2224 10$:     MovL    8(Sp), R2                  ; Make sure we have original R2            [15]
        41 62   18   E0 0EF8  2225          Bbs     #Cli$V_Adapter, -           ; Duplicates are OK if                     [15]
                        0EFC  2226                   Cli$L_Flags(R2), 60$       ; .. /Adapter was given                    [15]
        6E   08 A2   7D 0EFC  2227          MovQ    Cli$Q_File(R2), (Sp)       ; Set descriptor to device name            [15]
           6E   6E   3C 0F00  2228 20$:     MovZWL  (Sp), (Sp)                 ; Make sure length is word                 [15]
                53   D4 0F03  2229          ClrL    R3                         ; No matches so far                        [15]
55   00000000'EF   DE 0F05  2230          MovAL   Ds$GA_PTable, R5           ; Address of list                          [15]
              54   65   D0 0F0C  2231          MovL    (R5), R4                   ; Length of list                           [15]
           52   6544   D0 0F0F  2232 30$:     MovL    (R5)[R4], R2               ; Get address of P-tables                  [15]
                22   13 0F13  2233          BEql    50$                        ; Branch if no entry here                  [15]
     50   62   01   C3 0F15  2234          SubL3   #1, Hp$Q_Device(R2), R0    ; Length of name                           [15]
           6E   50   D1 0F19  2235          CmpL    R0, (Sp)                   ; Length match                             [15]
                19   12 0F1C  2236          BNeq    50$                        ; Branch if length mismatch                [15]
        51   04 AE   D0 0F1E  2237          MovL    4(Sp), R1                  ; Get address of desired                   [15]
  52   04 A2   01   C1 0F22  2238          AddL3   #1, Hp$Q_Device+4(R2),R2   ; Point to device name                     [15]
           81   82   91 0F27  2239 40$:     CmpB    (R2)+, (R1)+               ; Compare a byte                           [15]
                0B   12 0F2A  2240          BNeq    50$                        ; Branch if mismatch                       [15]
           F8 50   F5 0F2C  2241          SobGtr  R0, 40$                    ; Continue to compare                      [15]
                53   D5 0F2F  2242          TstL    R3                         ; Found a match...is it the first?         [15]
                12   12 0F31  2243          BNeq    80$                        ; No, generate an error                    [15]
        53   6544   D0 0F33  2244          MovL    (R5)[R4], R3               ; Get address of entry                     [15]
           D5 54   F5 0F37  2245 50$:     SobGtr  R4, 30$                    ; Next if any                              [15]
           B7 56   F4 0F3A  2246          SobGeq  R6, 10$                    ; If just did adapter, do device           [15]
        6E   01   D0 0F3D  2247 60$:     MovL    #1, (Sp)                   ; Success                                  [15]
        007F 8F   BA 0F40  2248 70$:     PopR    #^M<r0,r1,r2,r3,r4,r5,R6>  ;       ; Restore all registers             [15]
                05   0F44  2249          Rsb                                ; Return                                   [15]
                        0F45  2250
                6E   7F   0F45  2251 80$:     PushAQ  (Sp)                       ; Duplicate name descriptor on stack       [15]
     00000161'EF   9F 0F47  2252          PushAB  L^T_Ambig_Adap             ; Address of format string                 [15]
           07 56   E8 0F4D  2253          BlbS    R6, 90$                    ; Branch if this is adapter error          [15]
6E   0000018E'EF   9E 0F50  2254          MovAB   L^T_Ambig_Dev, (Sp)        ; Replace format string with device        [15]
                01   DD 0F57  2255 90$:     PushL   #Ds$K_Printf               ; Use PrintF typeout                       [15]
                15   DD 0F59  2256          PushL   #Ds$K_Type_Command_Err     ; Command error typecode                   [15]
00000000'GF     04   FB 0F5B  2257          CallS   #4, G^Dsx$Print            ; Print error                              [15]
                6E   D4 0F62  2258          ClrL    (Sp)                       ; Clear saved R0                           [15]
                DA   11 0F64  2259          Brb     70$                        ; Return                                   [15]
```

```
                        0F66  2261          .SBTTL  Locate /Adapter value
                        0F66  2262
                        0F66  2263 ;++
                        0F66  2264 ; Functional Description:
                        0F66  2265 ;
                        0F66  2266 ;       This routine uses the module-global Q_ADAPTER as the
                        0F66  2267 ;       name of an adapter device, if CLI$V_ADAPTER is set.
                        0F66  2268 ;       It will look up this device and return the address
                        0F66  2269 ;       of the Ptable.  If the device is the special name
                        0F66  2270 ;       'HUB', it will return 0.  Otherwise it will return
                        0F66  2271 ;       -1.
                        0F66  2272 ;
                        0F66  2273 ; Calling Sequence:
                        0F66  2274 ;       bsbw    loc$adapter
                        0F66  2275 ;
                        0F66  2276 ; Parameters: none
                        0F66  2277 ;
                        0F66  2278 ; Implicit inputs:
                        0F66  2279 ;       q_adapter       Descriptor of adapter name
                        0F66  2280 ;       cli$l_flags(r2) Checks CLI$V_ADAPTER bit.
                        0F66  2281 ;
                        0F66  2282 ; Outputs:
                        0F66  2283 ;       R6              Address of Ptable, or -1 if not found.
                        0F66  2284 ;
                        0F66  2285 ; Implicit inputs: none
                        0F66  2286 ;
                        0F66  2287 ; Return status:
                        0F66  2288 ;       R0      1 if succeeded, 0 if failed.  (Success includes no /Adapter).
                        0F66  2289 ;--
```

```
                                  0F66
                                  0F66  loc$adapter::                                      ;                                          [11]
             56     01     CE     0F66           mnegl    #1, r6                            ; Set to no /Adapter                       [11]
          4A 62     18     E1     0F69           bbc      #cli$v_adapter, -                 ; If not /Adapter                          [11]
                                  0F6D                     cli$l_flags(r2), 20$             ; ...then branch                           [11]
       50    00000002'EF   7D     0F6D           movq     q_adapter, r0                     ; Get descriptor of adapter device         [11]
             03     50     B1     0F74           cmpw     r0, #3                            ; Is length same as 'HUB'?                 [11]
                    0F     12     0F77           bneq     10$                               ; If not, try loc$ptable lookup            [11]
 00425548 8F   61   18     00     ED     0F79    cmpzv    #0, #24, (r1), #^A'HUB'           ; Compare with 'HUB' keyword               [11]
                    04     12     0F82           bneq     10$                               ; If not, try loc$ptable                   [11]
                    56     D4     0F84           clrl     r6                                ; OK, set return and go home               [11]
                    2F     11     0F86           brb      20$                               ; Return success                           [11]
                    04     BB     0F88  10$:     pushr    #^M<r2>                           ; Don't let R2 get wiped out               [11]
             52     01     CE     0F8A           mnegl    #1, r2                            ; Find it on any link                      [11]
                    2C     10     0F8D           bsbb     loc$ptable                        ; Find the Ptable for it                   [11]
                    04     BA     0F8F           popr     #^M<r2>                           ; And restore it                           [11]
             56     51     D0     0F91           movl     r1, r6                            ; Store address away                       [11]
                    21     12     0F94           bneq     20$                               ; Branch if found                          [11]
                 04 A2     D4     0F96           clrl     cli$l_command(r2)                 ; Otherwise, clear command address         [11]
       7E    00000002'EF   7D     0F99           movq     q_adapter, -(sp)                  ; Push adapter name desc on stack          [11]
             6E     6E     3C     0FA0           movzwl   (sp), (sp)                        ; Zero extend length, just in case         [11]
             00000124'EF   9F     0FA3           pushab   L^t_no_such_adapter               ; Message to type                          [11]
                    01     DD     0FA9           pushl    #ds$k_printf                      ; Use PRINTF                               [13]
                    15     DD     0FAB           pushl    #ds$k_type_command_err            ; Command error type code                  [13]
       00000000'GF  05     FB     0FAD           calls    #5, G^dsx$print                   ; Print it out                             [13]
                    50     D4     0FB4           clrl     r0                                ; Return failure                           [11]
                           05     0FB6           rsb                                        ; Return                                   [11]
             50     01     D0     0FB7  20$:     movl     #1, r0                            ; Set success                              [11]
                           05     0FBA           rsb                                        ; Return                                   [11]
```

```
                        .SUBTITLE       LOC$PTABLE Routine

;++
; FUNCTIONAL DESCRIPTION:
;
;       This routine locates a P-table given the device name.
;
; CALLING SEQUENCE:
;
;       BSBW    LOC$PTABLE
;
; INPUT PARAMETERS:     NONE
;
; IMPLICIT INPUTS:
;
;       R0      Length of name to be located
;       R1      Address of name to be located
;       R2      Link field of Ptable to locate
;
; OUTPUT PARAMETERS:    NONE
;
; IMPLICIT OUTPUTS:
;
;       R0      Index into DS$GA_PTABLE of located entry
;       R1      Address of P-table
;
; CONDITION CODES:
;
;       Z-bit   Clear if entry found, else set
;
; REGISTER USAGE:
;
;       R5      Address of DS$GA_PTABLE
;       R4      Current index into table
;--
```

```
                        OFBB  2357
                        OFBB  2358 LOC$PTABLE::
                3F  BB  OFBB  2359          pushr   #^M<r0,r1,r2,r3,r4,r5>  ; Save registers                          [09]
             6E 6E  3C  OFBD  2360          movzwl  (sp), (sp)              ; Make sure length is word                [11]
             53 52  D0  OFC0  2361          movl    r2, r3                  ; Do something with link address          [09]
                        OFC3  2362
         55 00000000'EF  DE  OFC3  2363          MOVAL   DS$GA_PTABLE,R5         ; Address of list
               54 65  D0  OFCA  2364          MOVL    (R5),R4                 ; Length of list
            52 6544  D0  OFCD  2365 10$:    MOVL    (R5)[R4],R2             ; Get address of P-tables
                2D  13  OFD1  2366          BEQL    40$                     ; Branch if no entry here
                53  D5  OFD3  2367          tstl    r3                      ; Is R2 valid, or wildcard?               [09]
                06  19  OFD5  2368          blss    20$                     ; If wildcard, continue compare           [09]
          20 A2 53  D1  OFD7  2369          cmpl    r3, hp$a_link(r2)       ; Compare link fields                     [09]
                23  12  OFDB  2370          bneq    40$                     ; Check next                              [09]
                        OFDD  2371 20$:                                     ;                                         [09]
       50 62 01  C3  OFDD  2372          SUBL3   #1,HP$Q_DEVICE(R2),R0   ; Length of name
             6E 50  D1  OFE1  2373          CMPL    R0,(SP)                 ; Length match
                1A  12  OFE4  2374          BNEQ    40$                     ; Branch if length mismatch
          51 04 AE  D0  OFE6  2375          MOVL    4(SP),R1                ; Get address of desired
       52 04 A2 01  C1  OFEA  2376          ADDL3   #1,HP$Q_DEVICE+4(R2),R2 ; Point to device name
             81 82  91  OFEF  2377 30$:    CMPB    (R2)+,(R1)+             ; Compare a byte
                0C  12  OFF2  2378          BNEQ    40$                     ; Branch if mismatch
             F8 50  F5  OFF4  2379          SOBGTR  R0,30$                  ; Continue to compare
            55 6544  D0  OFF7  2380          MOVL    (R5)[R4],R5             ; Get address of entry
             6E 54  7D  OFFB  2381          MOVQ    R4,(SP)                 ; Superceed saved R0/R1
                05  11  OFFE  2382          BRB     50$                     ; exit
             CA 54  F5  1000  2383 40$:    SOBGTR  R4,10$                  ; Next if any
                6E  7C  1003  2384          CLRQ    (SP)                    ; Clear R0/R1
                3F  BA  1005  2385 50$:    popr    #^M<r0,r1,r2,r3,r4,r5>  ; Restore all registers                   [09]
                51  D5  1007  2386          TSTL    R1                      ; Set condition codes from PT address
                    05  1009  2387          RSB
```

```
                     100A  2389          .SBTTL  LOC$PTDESC      Locate PT-descriptor
                     100A  2390
                     100A  2391 ;++
                     100A  2392 ; FUNCTIONAL DESCRIPTION:
                     100A  2393 ;
                     100A  2394 ;       This routine is used to locate the PT-descriptor
                     100A  2395 ;       given the hardware type name.
                     100A  2396 ;
                     100A  2397 ; CALLING SEQUENCE:
                     100A  2398 ;
                     100A  2399 ;       BSBW    LOC$PTDESC
                     100A  2400 ;
                     100A  2401 ; INPUT PARAMETERS:     NONE
                     100A  2402 ;
                     100A  2403 ; IMPLICIT INPUTS:
                     100A  2404 ;
                     100A  2405 ;       R0      Length of device type
                     100A  2406 ;       R1      Address of device type
                     100A  2407 ;
                     100A  2408 ; OUTPUT PARAMETERS:    NONE
                     100A  2409 ;
                     100A  2410 ; IMPLICIT OUTPUTS:
                     100A  2411 ;
                     100A  2412 ;       R0      Address of PT-descriptor
                     100A  2413 ;
                     100A  2414 ; CONDITION CODES:
                     100A  2415 ;
                     100A  2416 ;       Z-bit   Clear if found else set
                     100A  2417 ;
                     100A  2418 ; SIDE EFFECTS:         NONE
                     100A  2419 ;--
```

```
                                 100A  2421
                                 100A  2422 LOC$PTDESC::
                          3F BB  100A  2423          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save a couple
                                 100C  2424
                          52 D4  100C  2425          CLRL    R2                      ; Base Page 0/page 1
00000058 8F     0200    C2 D1  100E  2426          CMPL    L$L_HEADLENGTH(R2),#^X58; Check length of Header
                          0B 12  1017  2427          BNEQ    10$                     ; Branch if bad header
            55  021C    C2 D0  1019  2428          MOVL    L$A_DEVP(R2),R5         ; Address of DEVTYP list
                          04 13  101E  2429          BEQL    10$                     ; Branch if null
                          13 10  1020  2430          BSBB    PT_LOCATE               ; Find PT-descriptor here
                          09 12  1022  2431          BNEQ    20$                     ; Exit if found
                                 1024  2432 10$:
         55  00000000'EF DE  1024  2433          MOVAL   DS$GA_PTDESC,R5         ; Point to descriptor list
                          08 10  102B  2434          BSBB    PT_LOCATE               ; Search this list
                                 102D  2435 20$:
                       5E 08 C0  102D  2436          ADDL    #8,SP                   ; Remove saved R0/R1
                          3C BA  1030  2437          POPR    #^M<R2,R3,R4,R5>
                          50 D5  1032  2438          TSTL    R0                      ; Set condition codes from return
                          05 05  1034  2439          RSB                             ; Return
                                 1035  2440
                                 1035  2441 PT_LOCATE:
                       54 65 D0  1035  2442          MOVL    (R5),R4                 ; Get length of table
                          27 13  1038  2443          beql    40$                     ; Branch if table is empty              [08]
                    53  6544 D0  103A  2444 10$:     MOVL    (R5)[R4],R3             ; Get address of PT-descriptor
                 50  04 AE 7D  103E  2445          MOVQ    4(SP),R0                ; Get length,address
                    50 50 3C  1042  2446          movzwl  r0, r0                  ; Zero extend word length               [11]
                    83 50 91  1045  2447          CMPB    R0,(R3)+                ; Length match?
                          14 12  1048  2448          BNEQ    30$                     ; No match, next
                       83 81 91  104A  2449 20$:     CMPB    (R1)+,(R3)+             ; Compare byte
                          0F 12  104D  2450          BNEQ    30$                     ; Branch if not found
                       F8 50 F5  104F  2451          SOBGTR  R0,20$                  ; Count and branch
                 04 A3    80 8F 91  1052  2452          CMPB    #PD$_START,4(R3)        ; Verify PT-descriptor follows
                          05 12  1057  2453          BNEQ    30$                     ; Branch if not
                    50  6544 D0  1059  2454          MOVL    (R5)[R4],R0             ; Address of PT-descriptor
                          05 05  105D  2455          RSB                             ; Return with Z-bit clear
                                 105E  2456
                       D9 54 F5  105E  2457 30$:     SOBGTR  R4,10$                  ; Do next entry
                          50 D4  1061  2458 40$:     clrl    r0                      ; Not found                             [08]
                          05 05  1063  2459          RSB                             ; Return Z-bit Set
```

```
                                1064  2461          .SBTTL  INS$PTABLE      Insert P-table
                                1064  2462
                                1064  2463 ;++
                                1064  2464 ; FUNCTIONAL DESCRIPTION:
                                1064  2465 ;
                                1064  2466 ;       This routine can be called with the address of the P-table
                                1064  2467 ;       to be entered in DS$GA_PTABLE. It takes care of superceding
                                1064  2468 ;       existing Ptables for the same device name and link.
                                1064  2469 ;
                                1064  2470 ; CALLING SEQUENCE:
                                1064  2471 ;
                                1064  2472 ;       INS$PTABLE(PT)
                                1064  2473 ;
                                1064  2474 ; INPUT PARAMETERS:
                                1064  2475 ;
                                1064  2476 ;       AP      Address of P-table to insert
                                1064  2477 ;
                                1064  2478 ; IMPLICIT INPUTS:
                                1064  2479 ;
                                1064  2480 ;       DS$GA_PTABLE
                                1064  2481 ;
                                1064  2482 ; OUTPUT PARAMETERS:    NONE
                                1064  2483 ;
                                1064  2484 ; IMPLICIT OUTPUTS:
                                1064  2485 ;
                                1064  2486 ;       DS$GA_PTABLE
                                1064  2487 ;
                                1064  2488 ; COMPLETION CODES:     NONE
                                1064  2489 ;--
```

```
                                 1064  2491
                          003C   1064  2492 .ENTRY  INS$PTABLE,^M<R2,R3,R4,R5>
                       50  6C     7D  1066  2493          MOVQ    HP$Q_DEVICE(AP),R0      ; Get length/address of device name
                           51     D6  1069  2494          INCL    R1                      ; Skip "("
                           50     D7  106B  2495          DECL    R0                      ; Skip "-"
                     52    20 AC   D0  106D  2496          movl    hp$a_link(ap), r2       ; Link of device                        [09]
                 FFFFFF44 EF     16  1071  2497          Jsb     L^LOC$PTABLE            ; See if already exists                 [18]
                           48     13  1077  2498          BEQL    30$                     ; Branch if not found
                                      1079  2499
                55   00000000'EF   DE  1079  2500          MOVAL   DS$GA_PTABLE,R5         ; Get address of possibles
                       54    65     D0  1080  2501          MOVL    (R5),R4                 ; Get number of possibles
                     53   6544     D0  1083  2502 10$:     MOVL    (R5)[R4],R3             ; Point to this P-table
                           26     13  1087  2503          BEQL    20$                     ; Branch if none
                    20 A3   51     D1  1089  2504          CMPL    R1,HP$A_LINK(R3)        ; This P-table point to OLD PT for new
                           20     12  108D  2505          BNEQ    20$                     ; Branch if not
                 26 A3   5205  8F   B1  108F  2506          CMPW    #5!<^A'R'@8>,HP$T_TYPE(R3); Check for RB730. Special case.     [20]
                           0A     12  1095  2507          BNEQ    12$                     ; Branch if no match on first part.     [20]
              28 A3   30333742 8F   D1  1097  2508          CMPL    #^A'B730',2+HP$T_TYPE(R3) ; Finish checking                     [20]
                           0A     13  109F  2509          BEQL    15$                     ; Skip if match. Do not change PTABLE   [20]
                 18 A3    1C A1   CA  10A1  2510 12$:     BICL    HP$A_DVA(R1),HP$A_DEVICE(R3) ; Remove old DVA bits
                 18 A3    1C AC   C8  10A6  2511          BISL    HP$A_DVA(AP),HP$A_DEVICE(R3); Insert new DVA bits
                    20 A3    5C   D0  10AB  2512 15$:     MOVL    AP,HP$A_LINK(R3)        ; Point to new P-table
                         D1 54    F5  10AF  2513 20$:     SOBGTR  R4,10$                  ; Branch if more to process
                       6540    5C   D0  10B2  2514          MOVL    AP,(R5)[R0]             ; Use this index again
                         50    51   D0  10B6  2515          MOVL    R1,R0                   ; Copy address of dynamic memory
                   00000000'EF   16  10B9  2516          Jsb     L^EXE$DEANONPAGED       ; deallocate old PT                     [18]
                           2F     11  10BF  2517          BRB     60$                     ; Return normal
                                      10C1  2518 30$:
                55   00000000'EF   DE  10C1  2519          MOVAL   DS$GA_PTABLE,R5         ; Point to P-table list
                       54    65     D0  10C8  2520          MOVL    (R5),R4                 ; Get length of list
                           6544     D5  10CB  2521 40$:     TSTL    (R5)[R4]                ; Occupied?
                             1C     13  10CE  2522          BEQL    50$                     ; No, use it
                          F8 54     F5  10D0  2523          SOBGTR  R4,40$                  ; Search whole list
                                      10D3  2524          ERRSUP_S
                50   00660002 8F   D0  10E4  2525          MOVL    #DS$_ERROR,R0           ; Flag error
                                04    10FB  2526          RET                             ; Return
                                      10EC  2527 50$:
                         6544    5C   D0  10EC  2528          MOVL    AP,(R5)[R4]             ; Insert pointer to PT
             00 00000000'EF    54   E5  10F0  2529 60$:     BBCC    R4,DS$GA_SELECTS,70$    ; Clear select bit
                         50    01   D0  10F8  2530 70$:     MOVL    S^#SS$_NORMAL,R0        ; Success
                                04    10FB  2531          RET
                                      10FC  2532
                                      10FC  2533          .END
```

```
$$N                             =  00000002     D
$$T1                            =  00000001     D
$ER                             =  00000003     D
$MODULE                            00000000 R   D   02
ABORT_ATTACH                       00000001 R   D   03
ATC$_CMDLINE                       00000004     D
ATC$_PROMPT                        00000008     D
BADPARAM                           00000660 R   D   04
BAD_LINK                           000001BF R   D   04
BAD_NAME                           000002BF R   D   04
BAD_TYPE                           00000130 R   D   04
BAD_UNIT_EXC                       000002A2 R   D   04
BIT...                          =  00000004     D
BUFFER                             FFFFFF70     D
CENTRAL_BUS                        0000C2C7 R   D   02
CHECKAMBIGUOUS                     00000EDF R   D   04
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_SIZE                         00000444     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_LAST                         00000024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEST                         00000020     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       0000003C     D
CLI$V_ADAPTER                   =  00000018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BREAK                     =  000C000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
COMM_FLAGS                         00000000 R   D   03
CRD$M_CRD_DEBUG                 =  00000001     D
CRD$M_CTRL_C_NO_ECHO            =  00000002     D
CRD$M_FLUSH_CTRL_C              =  00000004     D
CRD$M_INHIBIT_ALLOCATE          =  00000008     D
CRD$V_CRD_DEBUG                 =  00000000     D
CRD$V_CTRL_C_NO_ECHO            =  00000001     D
CRD$V_FLUSH_CTRL_C              =  00000002     D
CRD$V_INHIBIT_ALLOCATE          =  00000003     D
DESC                               FFFFFFF4     D
DIR...                          =  00000001     D
DS$CNTRLC                          ********     X   00
DS$GA_PTABLE                       ********     X   00
DS$GA_PTDESC                       ********     X   00
DS$GA_SELECTS                      ********     X   00
DS$GB_INHIBIT_NAMING               ********     X   00
DS$GL_CLIBASE                      ********     X   00
DS$GL_CRD_FLAGS                    ********     X   00
DS$GL_FLAGS                        ********     X   00
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_PRINTB                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  0000001B     D
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        =  0000000B     D
DS$K_TYPE_FIRST_PASS            =  00000011     D
DS$K_TYPE_GENERAL               =  00000000     D
DS$K_TYPE_GENERAL_ERROR         =  00000003     D
DS$K_TYPE_NO_TESTS              =  00000012     D
```

```
DS$K_TYPE_PARAM_ERROR           = 0000001C     D        DS$V_EXCEPT                     = 00000013     D
DS$K_TYPE_PROGRAM_END           = 00000010     D        DS$V_EXETST                     = 00000012     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D        DS$V_HLTFLG                     = 00000003     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D        DS$V_LODFLG                     = 00000001     D
DS$K_TYPE_QIO_INVADP            = 00000024     D        DS$V_MEMMGT                     = 0000000F     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D        DS$V_OUTPUT                     = 00000017     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D        DS$V_RUBFLG                     = 00000005     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D        DS$V_SCRIPT                     = 00000015     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D        DS$V_SETIMR                     = 00000019     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D        DS$V_STRFLG                     = 00000002     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D        DS$V_SUBT                       = 0000000E     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D        DS$V_SYSFLG                     = 0000000A     D
DS$K_TYPE_START_ERR             = 00000018     D        DS$V_TIMRON                     = 00000011     D
DS$K_TYPE_START_LIST            = 00000025     D        DS$_ARITH                       = 006600D0     D
DS$K_TYPE_SUMMARY               = 0000000E     D        DS$_ASBE                        = 00660118     D
DS$K_TYPE_USER_PROMPT           = 00000002     D        DS$_BADLINK                     = 006600F0     D
DS$K_WARNING                    = 00000000     D        DS$_BADTYPE                     = 006600E8     D
DS$M_ABRTFLG                    = 00000040     D        DS$_BIIC                        = 00660120     D
DS$M_BADTIME                    = 00100000     D        DS$_CHME                        = 006600A8     D
DS$M_BATCH                      = 00400000     D        DS$_CHMK                        = 006600E0     D
DS$M_BRKCLR                     = 00001000     D        DS$_DEVNAME                     = 00660108     D
DS$M_BRKPT                      = 00000800     D        DS$_ERROR                       = 00660002     D
DS$M_CHARFLG                    = 00000100     D        DS$_FHWE                        = 00660068     D
DS$M_CMDFLG                     = 00000080     D        DS$_FRAGBUF                     = 00660080     D
DS$M_CTRLC                      = 00000001     D        DS$_ICBUSY                      = 006600C8     D
DS$M_CTRLO                      = 00010000     D        DS$_ICERR                       = 006600C0     D
DS$M_DEVFLG                     = 00000200     D        DS$_IHWE                        = 00660060     D
DS$M_DISABLCC                   = 01000000     D        DS$_ILLCHAR                     = 00660018     D
DS$M_DONFLG                     = 00002000     D        DS$_ILLPAGCNT                   = 00660078     D
DS$M_ERRFLG                     = 00000010     D        DS$_ILLUNIT                     = 00660100     D
DS$M_EXCEPT                     = 00080000     D        DS$_INSFMEM                     = 00660050     D
DS$M_EXETST                     = 00040000     D        DS$_IPL2HI                      = 006600B8     D
DS$M_HLTFLG                     = 00000008     D        DS$_IVADDR                      = 00660040     D
DS$M_LODFLG                     = 00000002     D        DS$_IVVECT                      = 00660038     D
DS$M_MEMMGT                     = 00008000     D        DS$_KRNLSTK                     = 00660090     D
DS$M_OUTPUT                     = 00800000     D        DS$_LOGIC                       = 00660070     D
DS$M_RUBFLG                     = 00000020     D        DS$_MCHK                        = 00660088     D
DS$M_SCRIPT                     = 00200000     D        DS$_MMOFF                       = 00660058     D
DS$M_SETIMR                     = 02000000     D        DS$_NEEDUNIT                    = 006600F8     D
DS$M_STRFLG                     = 00000004     D        DS$_NODE                        = 00660128     D
DS$M_SUBT                       = 00004000     D        DS$_NOPCS                       = 00660110     D
DS$M_SYSFLG                     = 00000400     D        DS$_NORMAL                      = 00660001     D
DS$M_TIMRON                     = 00020000     D        DS$_NOSUPPORT                   = 006600B1     D
DS$PRINTF                         ********     X    00  DS$_NOTDON                      = 00660030     D
DS$V_ABRTFLG                    = 0C000006     D        DS$_NOTIMP                      = 006600B0     D
DS$V_BADTIME                    = 00000014     D        DS$_NULLSTR                     = 00660010     D
DS$V_BATCH                      = 00000016     D        DS$_OVERFLOW                    = 00660008     D
DS$V_BRKCLR                     = 0000000C     D        DS$_POWER                       = 00660098     D
DS$V_BRKPT                      = 0000000B     D        DS$_PROGERR                     = 00660020     D
DS$V_CHARFLG                    = 00000008     D        DS$_SEVERE                      = 00660004     D
DS$V_CMDFLG                     = 00000007     D        DS$_TRANSL                      = 006600A0     D
DS$V_CTRLC                      = 00000000     D        DS$_TRUNCATE                    = 00660028     D
DS$V_CTRLO                      = 00000010     D        DS$_UNEXPINT                    = 006600D8     D
DS$V_DEVFLG                     = 00000009     D        DS$_VASFULL                     = 00660048     D
DS$V_DISABLCC                   = 00000018     D        DS$_WARNING                     = 00660000     D
DS$V_DONFLG                     = 0000000D     D        DSA$AL_APTMAIL                    0000FE00     D
DS$V_ERRFLG                     = 00000004     D        DSA$AT_APTTXT                     0000FA00     D
```

```
DSA$GL_APTCOM                   0000FE04     D
DSA$GL_DEVLEN                   0000FE58     D
DSA$GL_ERRNO                    0000FE44     D
DSA$GL_EVENT                    0000FE48     D
DSA$GL_FLAGS                    0000FE00     D
DSA$GL_MSGTYP                   0000FE40     D
DSA$GL_PASSES                   0000FE08     D
DSA$GL_PASSNO                   0000FE54     D
DSA$GL_SECTNO                   0000FE10     D
DSA$GL_SID                      0000FE14     D
DSA$GL_SUBTNO                   0000FE4C     D
DSA$GL_TESTNO                   0000FE50     D
DSA$GL_UNITS                    0000FE0C     D
DSA$GQ_MSGPTR                   0000FE68     D
DSA$GT_DE.NAM                   0000FE5C     D
DSA$V_APT                   =   0000001F     D
DSA$V_USER                  =   0000001C     D
DSR$COMPLETION                  ********   X   00
DSR$IFLOADDEV                   ********   X   00
DSV$ATTACH                      00000000 RG  D   04
DSV$DEATTACH                    00000B37 RG  D   04
DSV$DESELECT                    00000D56 RG  D   04
DSV$SELECT                      00000BD0 RG  D   04
DSV$SHOWDEVICE                  00000AA5 RG  D   04
DSV$SHOWDEVICEB                 00000A9E RG  D   04
DSV$SHOWSELECT                  00000B06 RG  D   04
DSV$SHOWSELECTB                 00000AFF RG  D   04
DSX$ATTACH                      0000010A RG  D   04
DSX$PRINT                       ********   X   00
DS_ERRSUP                       ********   X   00
DS_SUPERLINE                    ********   X   00
EARG$_BASE                      00000008     D
EARG$_PC                        00000004     D
EARG$_TEXT                      0000000C     D
ERROR_EXIT                      00000410 R   D   04
EXE$ALONONPAGED                 ********   X   00
EXE$DEANONPAGED                 ********   X   00
EXTRACT                         000008F2 R   D   04
EX_ADD                          00000944 R   D   04
EX_BADPARAM                     00000913 R   D   04
EX_CASE                         00000948 R   D   04
EX_COMMON                       00000A23 R   D   04
EX_COMPLEMENT                   00000946 R   D   04
EX_DECIMAL                      00000919 R   D   04
EX_FETCH                        00000944 R   D   04
EX_FINISH                       0000095C R   D   04
EX_HEXADECIMAL                  00000919 R   D   04
EX_LITERAL                      00000944 R   D   04
EX_LOGICAL                      00000936 R   D   04
EX_NAME                         00000951 R   D   04
EX_OCTAL                        00000919 R   D   04
EX_START                        00000917 R   D   04
EX_STORE                        00000969 R   D   04
EX_STRING                       00000924 R   D   04
FLG$V_BRIEF                 =   00000000     D
FORMAT_PROMPT                   00000A75 R   D   04
FPT$_ARGS                       00000008     D
FPT$_FORMAT                     00000004     D
GENERIC                         FFFFFFEB     D
GET_ATTRIBUTES                  00000425 R   D   04
GET_LINE                        00000A4D R   D   04
GET_LINK                        000001DC R   D   04
GET_NAME                        0000037C R   D   04
GET_TYPE                        0000014D R   D   04
HP$A_DEPENDENT                  00000032     D
HP$A_DEVICE                     00000018     D
HP$A_DVA                        0000001C     D
HP$A_LINK                       00000020     D
HP$B_DRIVE                      0000000B     D
HP$B_FLAGS                      0000000A     D
HP$C_NAMELEN                =   00000014   G D
HP$M_NAME                   =   00000004     D
HP$M_WASALL                 =   00000002     D
HP$Q_DEVICE                     00000000     D
HP$T_DEVICE                     0000000C     D
HP$T_TYPE                       00000026     D
HP$V_ALLOC                  =   00000000     D
HP$V_NAME                   =   00000002     D
HP$V_WASALL                 =   00000001     D
HP$W_SIZE                       00000008     D
HP$W_VECTOR                     00000024     D
IARG$_BASE                      00000008     D
IARG$_PC                        00000004     D
IARG$_PROMPT                    00000010     D
IARG$_TEXT                      0000000C     D
IF_ALPHA                        000004C5 R   D   04
INI$PTABLE                      00000EC2 RG  D   04
INS$PTABLE                      00001064 RG  D   04
INTERPRET                       0000063B R   D   04
KB_CHECK                        ********   X   00
L$A_CCP                         00000240     D
L$A_DEVP                        0000021C     D
L$A_DREG                        00000224     D
L$A_DTP                         00000218     D
L$A_ICP                         0000023C     D
L$A_LASTADR                     00000214     D
L$A_NAME                        00000208     D
L$A_REPP                        00000244     D
L$A_SECNAM                      00000250     D
L$A_STATAB                      00000248     D
L$A_TSTCNT                      00000254     D
L$L_ENVIRON                     00000204     D
L$L_ERRTYP                      0000024C     D
L$L_HEADLENGTH                  00000200     D
L$L_REV                         0000020C     D
L$L_UNIT                        00000220     D
L$L_UNUSED                      00000228     D
L$L_UPDATE                      00000210     D
LOC$ADAPTER                     00000F66 RG  D   04
LOC$PTABLE                      00000FBB RG  D   04
LOC$PTDESC                      0000100A RG  D   04
LOCAL                           FFFFFF70     D
LOCAL_BLOCK                     FFFFFFE7     D
MAX_UNIT                        FFFFFFFD     D
```

```
NAME_FLAGS                      FFFFFFFF     D
OLDPC                           FFFFFFFC     D
PARENT_CONTROLLER               FFFFFFFE     D
PARENT_PTABLE                   FFFFFFF5     D
PARSE_DEVICE                    000004DA RG  D  04
PART_LIST                       000002A9 R   D  02
PD$_ADD                     =   0000008A     D
PD$_CASE                    =   0000008C     D
PD$_COMPLEMENT              =   00000089     D
PD$_DECIMAL                 =   00000082     D
PD$_END                     =   00000081     D
PD$_FETCH                   =   00000087     D
PD$_HEXADECIMAL             =   00000084     D
PD$_LITERAL                 =   00000086     D
PD$_LOGICAL                 =   0000008B     D
PD$_NAME                    =   0000008D     D
PD$_OCTAL                   =   00000083     D
PD$_START                   =   00000080     D
PD$_STORE                   =   00000088     D
PD$_STRING                  =   00000085     D
PD_ADD                          00000699 R   D  04
PD_CASE                         0000072B R   D  04
PD_COMPLEMENT                   00000694 R   D  04
PD_DECIMAL                      00000754 R   D  04
PD_FETCH                        00000672 R   D  04
PD_FINISH                       00000669 R   D  04
PD_HEXADECIMAL                  000007B5 R   D  04
PD_LITERAL                      0000066D R   D  04
PD_LOGICAL                      000006B3 R   D  04
PD_NAME                         00000748 R   D  04
PD_NUMERIC                      000007E9 R   D  04
PD_OCTAL                        00000785 R   D  04
PD_START                        00000664 R   D  04
PD_STORE                        00000683 R   D  04
PD_STRING                       00000851 R   D  04
PT$EXTRACT                      000008E0 RG  D  04
PT$INTERPRET                    00000612 RG  D  04
PTD$M_CONTROLLER            =   00000002     D
PTD$M_DEVICE                =   00000003     D
PTD$M_ENDDEVICE             =   0000001B     D
PTD$M_INHERIT               =   00000018     D
PTD$M_INHERITED             =   00000008     D
PTD$M_INHERIT_CON           =   00000010     D
PTD$M_INHERIT_PRE           =   00000008     D
PTD$M_NAME                  =   00000004     D
PTD$M_UNIT                  =   00000001     D
PTD$V_CONTROLLER            =   00000001     D
PTD$V_INHERIT_CON           =   00000004     D
PTD$V_INHERIT_PRE           =   00000003     D
PTD$V_NAME                  =   00000002     D
PTD$V_UNIT                  =   00000000     D
PTDESC                          FFFFFFF9     D
PT_LOCATE                       00001035 R   D  04
Q_ADAPTER                       00000002 RG  D  03
Q_HUB                           0000029E R   D  02
SAVABS...                   =   0000000C     D
SCAN$ALPHA                      ********   X   00
SCAN$ALPHANUM                   ********   X   00
SCAN$DECIMAL                    ********   X   00
SCAN$DEVICE                     ********   X   00
SCAN$NUMERIC                    ********   X   00
SCAN$SPACES                     ********   X   00
SCAN$SYMBOL                     ********   X   00
SCAN_ALPHA                      000005FA R   D  04
SCAN_ALPHANUM                   00000606 R   D  04
SCRIPT$FLUSH                    ********   X   00
SET_ABORT                       000000FD RG  D  04
SHOWBUF_SIZ                 =   00000100     D
SHOW_DEV                        00000E39 R   D  04
SIZ...                      =   00000001     D
SKIP                            0000093D R   D  04
SS$_BADPARAM                =   00000014     D
SS$_DEVALLOC                =   00000840     D
SS$_DEVALRALLOC             =   00000641     D
SS$_INSFARG                 =   00000114     D
SS$_NORMAL                  =   00000001     D
SS$_NOSUCHDEV               =   00000908     D
ST_DECIMAL                      000009AB R   D  04
ST_HEXADECIMAL                  000009BD R   D  04
ST_LOGICAL                      000009F9 R   D  04
ST_OCTAL                        000009AB R   D  04
ST_STRING                       000009CE R   D  04
SYS$ALLOC                       ********  GX   00
SYS$DALLOC                      ********  GX   00
SYS$FAO                         ********   X   00
SYS$FAOL                        ********  G    04
T_AMBIG_ADAP                    00000161 R   D  02
T_AMBIG_DEV                     0000018E R   D  02
T_BAD_NAME                      0000002E R   D  02
T_DEATTACH                      000001C5 R   D  02
T_DEVALLOC                      000001E1 R   D  02
T_INVALID                       00000007 R   D  02
T_LINK                          000000F6 R   D  02
T_MAIN_BUS                      000000A7 R   D  02
T_NAME                          000000EA R   D  02
T_NO_SUCH_ADAPTER               00000124 R   D  02
T_NO_SUCH_DEV                   0000010E R   D  02
T_NO_SUCH_ON_ADAPTER            0000013B R   D  02
T_PART_OF                       00000068 R   D  02
T_PNF                           000002?? R   D  02
T_PROMPT                        000000?? R   D  02
T_SHOULDBE                      00000269 R   D  02
T_SHOWDEV                       00000202 R   D  02
T_SHOWDEVB                      0000021C R   D  02
T_SKIP                          00000254 R   D  02
T_STRING                        00000281 R   D  02
T_TYPE                          000001C2 R   D  02
T_UNIT_EXCESS                   00000075 R   D  02
T_YESNO                         00000273 R   D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFF (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000002DE (734.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	0000000A (10.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	000010FC (4348.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000002	1777 (4)	1777 (4)
$$T1	=00000001	1932 (4)	1932 (4)
$ER	=00000003	2524 (6)	#-2524 (6) #-352 (1)
$MODULE	00000000-R	188 (1)	2524 (6) 352 (1)
ABORT_ATTACH	00000001-R	259 (1)	#-295 (1) #-333 (1) #-361 (1)
ATC$_CMDLINE	00000004	417 (1)	432 (1)
ATC$_PROMPT	00000008	417 (1)	437 (1)
BADPARAM	00000C60-R	1088 (3)	#-1070 (3)
BAD_LINK	000001BF-R	493 (1)	#-508 (1) #-554 (1)
BAD_NAME	000002BF-R	609 (1)	#-676 (1)
BAD_TYPE	00000130-R	446 (1)	#-460 (1) #-464 (1)
BAD_UNIT_EXC	000002A2-R	571 (1)	#-684 (1)
BIT...	=00000004	164 (1)	151 (1) 152 (1) 156 (1) 161 (1) 163 (1) 164 (1)
BUFFER	FFFFFF70	1054 (3)	1061 (3) 1268 (3)
CENTRAL_BUS	000002C7-R	244 (1)	519 (1)
CHECKAMBIGUOUS	00000EDF-R	2216 (6)	#-1319 (4) #-1992 (4)
CLI$K_BUFSIZ	=00000100	160 (1)	160 (1)
CLI$K_SIZE	00000444	160 (1)	
CLI$L_ADDRESS	00000018	160 (1)	
CLI$L_COMMAND	00000004	160 (1)	#-2309 (6)
CLI$L_DATA	0000001C	160 (1)	
CLI$L_FLAGS	00000000	160 (1)	2220 (6) 2226 (6) 2295 (6)
CLI$L_LAST	00000024	160 (1)	
CLI$L_NEXT	00000030	160 (1)	
CLI$L_PASS	0000002C	160 (1)	
CLI$L_SUBT	00000028	160 (1)	
CLI$L_TEST	00000020	160 (1)	
CLI$Q_BUFQWD	00000034	160 (1)	
CLI$Q_FILE	00000008	160 (1)	#-1618 (4) #-1739 (4) #-1746 (4) #-1826 (4) #-1872 (4) #-1879 (4) #-1999 (4) #-2039 (4) #-2046 (4) #-2227 (6) #-307 (1) 308 (1) #-310 (1) 313 (1) #-335 (1) 337 (1) #-339 (1) #-343 (1)
CLI$Q_SECTION	00000010	160 (1)	
CLI$Q_TIME	0000043C	160 (1)	
CLI$T_BUFFER	0000003C	160 (1)	
CLI$V_ADAPTER	=00000018	160 (1)	#-2219 (6) #-2225 (6) #-2294 (6)
CLI$V_ADR	=0000000B	160 (1)	
CLI$V_ASCII	=00000013	160 (1)	
CLI$V_BREAK	=0000000A	160 (1)	
CLI$V_BRIEF	=0000001B	160 (1)	
CLI$V_BYTE	=0000000D	160 (1)	
CLI$V_CLEAR	=00000002	160 (1)	
CLI$V_DEC	=00000010	160 (1)	
CLI$V_DEFAULT	=0000000C	160 (1)	
CLI$V_DEPOSIT	=00000019	160 (1)	
CLI$V_EVENT	=00000008	160 (1)	
CLI$V_EXAM	=00000005	160 (1)	
CLI$V_FLAGS	=00000009	160 (1)	

```
CLI$V_HEX                       =00000012      160     (1)
CLI$V_KERNEL                    =00000017      160     (1)
CLI$V_LOAD                      =00000006      160     (1)
CLI$V_LONG                      =0000000F      160     (1)
CLI$V_NOTNUF                    =00000001      160     (1)
CLI$V_OCT                       =00000011      160     (1)
CLI$V_PREG                      =0000001A      160     (1)
CLI$V_QA                        =00000007      160     (1)
CLI$V_QACKLOOPLOOPS             =0000001C      160     (1)
CLI$V_QAERRORPRINTS             =0000001B      160     (1)
CLI$V_QAMULTIPLEPASS            =0000001F      160     (1)
CLI$V_QASUBTESTLOOPS            =0000001E      160     (1)
CLI$V_QATESTLOOPS               =0000001D      160     (1)
CLI$V_REG                       =00000014      160     (1)
CLI$V_REQUIRED                  =00000C00      160     (1)
CLI$V_RUN                       =00000015      160     (1)
CLI$V_SET                       =00000003      160     (1)
CLI$V_SHOW                      =00000004      160     (1)
CLI$V_VALSEC                    =00000016      160     (1)
CLI$V_WORD                      =0000000E      160     (1)
COMM_FLAGS                       00000000-R    257     (1)    #-1616  (4)   #-1648  (4)   #-1686  (4)   #-1700  (4)
                                                                2129  (6)
CRD$M_CRD_DEBUG                 =00000001      164     (1)
CRD$M_CTRL_C_NO_ECHO            =00000002      164     (1)
CRD$M_FLUSH_CTRL_C              =00000004      164     (1)
CRD$M_INHIBIT_ALLOCATE          =00000008      164     (1)
CRD$V_CRD_DEBUG                 =00000000      164     (1)
CRD$V_CTRL_C_NO_ECHO            =00000001      164     (1)
CRD$V_FLUSH_CTRL_C              =00000002      164     (1)
CRD$V_INHIBIT_ALLOCATE          =00000003      164     (1)    #-1925  (4)
DESC                             FFFFFFF4      1054    (3)    #-1061  (3)   #-1282  (3)     1284  (3)
DIR...                          =00000001      1564    (4)      1048  (3)     1054  (3)     1334  (3)     1564  (4)
                                                                 417  (1)      427  (1)
DS$CNTRLC                        00000000-XR                     181  (1)      296  (1)      348  (1)
DS$GA_PTABLE                     00000000-XR                    1621  (4)     1690  (4)     1736  (4)     177   (1)
                                                                1821  (4)     1994  (4)     2181  (6)     2230  (6)
                                                                2363  (6)     2500  (6)     2519  (6)
DS$GA_PTDESC                     00000000-XR                    177   (1)     2433  (6)
DS$GA_SELECTS                    00000000-XR                    1693  (4)      171  (1)     1780  (4)     1822  (4)
                                                                1995  (4)     2184  (6)     2529  (6)
DS$GB_INHIBIT_NAMING             00000000-XR                    181   (1)    #-797  (3)
DS$GL_CLIBASE                    00000000-XR                    171   (1)
DS$GL_CRD_FLAGS                  00000000-XR                    182   (1)     1925  (4)
DS$GL_FLAGS                      00000000-XR                    179   (1)      319  (1)
DS$K_ERROR                      =00000002      156     (1)
DS$K_NORMAL                     =00000001      156     (1)
DS$K_PRINTB                     =00000002      163     (1)
DS$K_PRINTF                     =00000001      163     (1)    #-1749  (4)   #-1777  (4)   #-1875  (4)   #-1882  (4)
                                                              #-1953  (4)   #-2042  (4)   #-2049  (4)   #-2138  (6)
                                                              #-2147  (6)   #-2255  (6)   #-2313  (6)   #-322   (1)
                                                              #-326   (1)
DS$K_PRINTI                     =00000000      163     (1)
DS$K_PRINTX                     =00000003      163     (1)
DS$K_SEVERE                     =00000004      156     (1)
DS$K_SUBSYS                     =00000066      156     (1)      156  (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014      163     (1)
DS$K_TYPE_ABORT_TEST            =00000013      163     (1)
```

```
DS$K_TYPE_COMMAND_ERR           =00000015     163    (1)    #-1750   (4)    #-1876   (4)    #-1883   (4)    #-1954   (4)
                                                            #-2043   (4)    #-2050   (4)    #-2256   (6)    #-2314   (6)
                                                            #-323    (1)    #-327    (1)
DS$K_TYPE_COMMAND_OUT           =00000016     163    (1)    #-1777   (4)    #-2139   (6)    #-2148   (6)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     163    (1)
DS$K_TYPE_DS_PROMPT             =00000001     163    (1)
DS$K_TYPE_DS_START              =0U000001D    163    (1)
DS$K_TYPE_ERRDEV                =00000008     163    (1)
DS$K_TYPE_ERRHARD               =00000006     163    (1)
DS$K_TYPE_ERROR_BODY            =00000009     163    (1)
DS$K_TYPE_ERROR_END             =0000000A     163    (1)
DS$K_TYPE_ERRPREP               =0000001B     163    (1)
DS$K_TYPE_ERRSOFT               =00000007     163    (1)
DS$K_TYPE_ERRSUP                =00000004     163    (1)
DS$K_TYPE_ERRSYS                =00000C05     163    (1)
DS$K_TYPE_ERR_HALT              =0000000D     163    (1)
DS$K_TYPE_EXCEPTION             =0000000C     163    (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     163    (1)
DS$K_TYPE_FIRST_PASS            =00000011     163    (1)
DS$K_TYPE_GENERAL               =00000000     163    (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     163    (1)
DS$K_TYPE_NO_TESTS              =00000012     163    (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     163    (1)
DS$K_TYPE_PROGRAM_END           =00000010     163    (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     163    (1)
DS$K_TYPE_PROGRAM_START         =0000000F     163    (1)
DS$K_TYPE_QIO_INVADP            =00000024     163    (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     163    (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     163    (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     163    (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     163    (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     163    (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     163    (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     163    (1)
DS$K_TYPE_START_ERR             =00000018     163    (1)
DS$K_TYPE_START_LIST            =00000025     163    (1)
DS$K_TYPE_SUMMARY               =0000000E     163    (1)
DS$K_TYPE_USER_PROMPT           =00000002     163    (1)
DS$K_WARNING                    =00000000     156    (1)
DS$M_ABRTFLG                    =00000040     161    (1)
DS$M_BADTIME                    =00100000     161    (1)
DS$M_BATCH                      =00400000     161    (1)
DS$M_BRKCLR                     =00001000     161    (1)
DS$M_BRKPT                      =00000800     161    (1)
DS$M_CHARFLG                    =00000100     161    (1)
DS$M_CMDFLG                     =00000080     161    (1)
DS$M_CTRLC                      =00000001     161    (1)
DS$M_CTRLO                      =00010000     161    (1)
DS$M_DEVFLG                     =00000200     161    (1)
DS$M_DISABLCC                   =01000000     161    (1)
DS$M_DONFLG                     =00002000     161    (1)
DS$M_ERRFLG                     =00000010     161    (1)
DS$M_EXCEPT                     =00080000     161    (1)
DS$M_EXETST                     =00040000     161    (1)
DS$M_HLTFLG                     =00000008     161    (1)
DS$M_LODFLG                     =00000002     161    (1)
DS$M_MEMMGT                     =00008000     161    (1)
```

```
DS$M_OUTPUT                     =00800000      161     (1)
DS$M_RUBFLG                     =00000020      161     (1)
DS$M_SCRIPT                     =00200000      161     (1)
DS$M_SETIMR                     =02000000      161     (1)
DS$M_STRFLG                     =00000004      161     (1)
DS$M_SUBT                       =00004000      161     (1)
DS$M_SYSFLG                     =00000400      161     (1)
DS$M_TIMRON                     =00020000      161     (1)
DS$PRINTF                        00000000-XR                    172     (1)
DS$V_ABRTFLG                    =00000006      161     (1)
DS$V_BADTIME                    =00000014      161     (1)
DS$V_BATCH                      =00000016      161     (1)
DS$V_BRKCLR                     =0000000C      161     (1)
DS$V_BRKPT                      =0000000B      161     (1)
DS$V_CHARFLG                    =00000008      161     (1)
DS$V_CMDFLG                     =00000007      161     (1)
DS$V_CTRLC                      =00000000      161     (1)
DS$V_CTRLO                      =00000010      161     (1)
DS$V_DEVFLG                     =00000009      161     (1)
DS$V_DISABLCC                   =00000018      161     (1)
DS$V_DONFLG                     =0000000D      161     (1)
DS$V_ERRFLG                     =00000004      161     (1)
DS$V_EXCEPT                     =00000013      161     (1)
DS$V_EXETST                     =00000012      161     (1)
DS$V_HLTFLG                     =00000003      161     (1)
DS$V_LODFLG                     =00000001      161     (1)
DS$V_MEMMGT                     =0000000F      161     (1)
DS$V_OUTPUT                     =00000017      161     (1)
DS$V_RUBFLG                     =00000005      161     (1)
DS$V_SCRIPT                     =00000015      161     (1)     #-318     (1)
DS$V_SETIMR                     =00000019      161     (1)
DS$V_STRFLG                     =00000002      161     (1)
DS$V_SUBT                       =0000000E      161     (1)
DS$V_SYSFLG                     =0000000A      161     (1)
DS$V_TIMRON                     =00000011      161     (1)
DS$_ARITH                       =006600D0      156     (1)
DS$_ASBE                        =00660118      156     (1)
DS$_BADLINK                     =006600F0      156     (1)     #-497     (1)     #-541     (1)
DS$_BADTYPE                     =006600E8      156     (1)     #-450     (1)
DS$_BIIC                        =00660120      156     (1)
DS$_CHME                        =006600A8      156     (1)
DS$_CHMK                        =006600E0      156     (1)
DS$_DEVNAME                     =00660108      156     (1)     #-661     (1)
DS$_ERROR                       =00660002      156     (1)     #-2525    (6)
DS$_FHWE                        =00660068      156     (1)
DS$_FRAGBUF                     =00660080      156     (1)
DS$_ICBUSY                      =006600C8      156     (1)
DS$_ICERR                       =006600C0      156     (1)
DS$_IHWE                        =00660060      156     (1)
DS$_ILLCHAR                     =00660018      156     (1)
DS$_ILLPAGCNT                   =00660078      156     (1)
DS$_ILLUNIT                     =00660100      156     (1)     #-575     (1)
DS$_INSFMEM                     =00660050      156     (1)
DS$_IPL2HI                      =006600B8      156     (1)
DS$_IVADDR                      =00660040      156     (1)
DS$_IVVECT                      =00660038      156     (1)
DS$_KRNLSTK                     =00660090      156     (1)
```

```
DS$_LOGIC                       =00660C70      156     (1)
DS$_MCHK                        =00660088      156     (1)
DS$_MMOFF                       =00660058      156     (1)
DS$_NEEDUNIT                    =006600F8      156     (1)
DS$_NODE                        =00660128      156     (1)
DS$_NOPCS                       =00660110      156     (1)
DS$_NORMAL                      =00660001      156     (1)
DS$_NOSUPPORT                   =006600B1      156     (1)
DS$_NOTDON                      =00660030      156     (1)
DS$_NOTIMP                      =006600B0      156     (1)       156     (1)
DS$_NULLSTR                     =00660010      156     (1)
DS$_OVERFLOW                    =00660008      156     (1)
DS$_POWER                       =00660098      156     (1)
DS$_PROGERR                     =00660020      156     (1)
DS$_SEVERE                      =00660C04      156     (1)
DS$_TRANSL                      =006600A0      156     (1)
DS$_TRUNCATE                    =00660028      156     (1)
DS$_UNEXPINT                    =006600D8      156     (1)
DS$_VASFULL                     =00660048      156     (1)
DS$_WARNING                     =00660000      156     (1)       156     (1)
DSA$GL_FLAGS                     0000FE00                        1924    (4)       2067    (4)       317     (1)
DSA$GL_SID                       0000FE14                      #-534     (1)
DSA$V_APT                       =0000001F                      #-316     (1)
DSA$V_USER                      =0000001C                      #-1923    (4)     #-2066    (4)
DSR$COMPLETION                   00000000-XR                     178     (1)       1950    (4)
DSR$IFLOADDEV                    00000000-XR                     178     (1)       1847    (4)
DSV$ATTACH                       00000000-R     294     (1)
DSV$DEATTACH                     00000B37-R     1733    (4)
DSV$DESELECT                     00000D56-R     1989    (4)
DSV$SELECT                       00000BD0-R     1816    (4)
DSV$SHOWDEVICE                   00000AA5-R     1617    (4)
DSV$SHOWDEVICEB                  00000A9E-R     1615    (4)
DSV$SHOWSELECT                   00000B06-R     1687    (4)
DSV$SHOWSELECTB                  00000AFF-R     1685    (4)
DSX$ATTACH                       0000010A-R     429     (1)       314     (1)
DSX$PRINT                        00000000-XR                     173     (1)       1751    (4)       1777    (4)       181     (1)
                                                                 1877    (4)       1884    (4)       1955    (4)       2044    (4)
                                                                 2051    (4)       2140    (6)       2149    (6)       2257    (6)
                                                                 2315    (6)       324     (1)       328     (1)
DS_ERRSUP                        00000000-XR                     172     (1)       2524    (6)       352     (1)
DS_SUPERLINE                     00000000-XR                     172     (1)       341     (1)
EARG$_BASE                       00000008       1334    (3)
EARG$_PC                         00000004       1334    (3)     #-1338    (3)
EARG$_TEXT                       0000000C       1334    (3)     #-1339    (3)       #-1406    (3)       #-1407    (3)
ERROR_EXIT                       00000410-R     718     (2)     #-451     (1)       #-476     (1)       #-498     (1)       #-504     (1)
                                                                 #-542     (1)       #-576     (1)       #-663     (1)       #-672     (1)
                                                                 #-713     (2)       #-715     (2)
EXE$ALONONPAGED                  00000000-XR                     173     (1)       473     (1)
EXE$DEANONPAGED                  00000000-XR                     173     (1)       1782    (4)       2516    (6)       724     (3)
EXTRACT                          000008F2-R     1343    (3)     #-1364    (3)       #-1371    (3)       #-1380    (3)       1383    (3)
                                                                 #-1394    (3)       #-1398    (3)       #-1403    (3)       #-1419    (3)
                                                                 #-1427    (4)       #-1479    (4)
EX_ADD                           00000944-R     1391    (3)     1358      (3)
EX_BADPARAM                      00000913-R     1359    (3)     #-1342    (3)
EX_CASE                          00000948-R     1395    (3)     1358      (3)
EX_COMMON                        00000A23-R     1464    (4)     #-1430    (4)       #-1434    (4)       #-1438    (4)       #-1448    (4)
                                                                 #-1450    (4)       #-1459    (4)
```

```
EX_COMPLEMENT                   00000946-R  1393  (3)    1358  (3)
EX_DECIMAL                      00000919-R  1365  (3)    1358  (3)
EX_FETCH                        00000944-R  1390  (3)    1358  (3)
EX_FINISH                       0000095C-R  1405  (3)    1358  (3)
EX_HEXADECIMAL                  00000919-R  1367  (3)    1358  (3)
EX_LITERAL                      00000944-R  1389  (3)    1358  (3)
EX_LOGICAL                      00000936-R  1381  (3)    1358  (3)
EX_NAME                         00000951-R  1400  (3)    1358  (3)
EX_OCTAL                        00000919-R  1366  (3)    1358  (3)
EX_START                        00000917-R  1363  (3)    1358  (3)
EX_STORE                        00000969-R  1412  (3)    1358  (3)
EX_STRING                       00000924-R  1372  (3)    1358  (3)
FLG$V_BRIEF                     =00000000    256  (1)  #-1616  (4)  #-1648  (4)  #-1686  (4)  #-1700  (4)
                                                        #-2129  (6)
FORMAT_PROMPT                   00000A75-R  1566  (4)    1164  (3)    1230  (3)    1287  (3)    1527  (4)
                                                          449  (1)     496  (1)     540  (1)     574  (1)
                                                          660  (1)
FPT$_ARGS                       00000008    1564  (4)    1572  (4)
FPT$_FORMAT                     00000004    1564  (4)    1572  (4)
GENERIC                         FFFFFFEB     427  (1)     640  (1)  #-753  (3)  #-769  (3)     772  (3)
                                                        #-793  (3)     796  (3)  #-799  (3)     873  (3)
GET_ATTRIBUTES                  00000425-R   750  (3)  #-668  (1)
GET_LINE                        00000A4D-R  1517  (4)    1134  (3)    1201  (3)    1239  (3)     455  (1)
                                                          502  (1)     670  (1)
GET_LINK                        000001DC-R   500  (1)  #-492  (1)
GET_NAME                        0000037C-R   667  (1)  #-546  (1)  #-567  (1)
GET_TYPE                        0000014D-R   453  (1)  #-444  (1)
HP$A_DEVICE                     00000018                #-2131  (6)  #-2510  (6)  #-2511  (6)  #-557  (1)
HP$A_DVA                        0000001C                #-2510  (6)  #-2511  (6)  #-557  (1)
HP$A_LINK                       00000020                #-1629  (4)  #-1770  (4)  #-1842  (4)  #-2016  (4)
                                                          2132  (6)  #-2369  (6)  #-2496  (6)  #-2504  (6)
                                                        #-2512  (6)  #-556  (1)
HP$B_DRIVE                      0000000B                #-685  (1)
HP$B_FLAGS                      0000000A                  1929  (4)    1931  (4)    1936  (4)  #-1939  (4)
                                                          2070  (4)    2072  (4)     649  (1)     688  (1)
                                                        #-865  (3)
HP$C_NAMELEN                    =00000014    183  (1)  #-472  (1)  #-690  (1)
HP$M_NAME                       =00000004               #-865  (3)
HP$M_WASALL                     =00000002               #-1938  (4)
HP$Q_DEVICE                      00000000               #-1634  (4)  #-1844  (4)    1932  (4)  #-1941  (4)
                                                          2073  (4)    2136  (6)    2145  (6)  #-2234  (6)
                                                        #-2238  (6)  #-2372  (6)  #-2376  (6)  #-2493  (6)
                                                        #-696  (1)  #-697  (1)  #-778  (3)  #-780  (3)
HP$T_DEVICE                     0000000C                #-563  (1)    694  (1)     959  (3)
HP$T_TYPE                       00000026                  2114  (6)    2135  (6)    2144  (6)  #-2506  (6)
                                                        #-2508  (6)    484  (1)
HP$V_ALLOC                      =00000000               #-1928  (4)  #-2069  (4)
HP$V_NAME                       =00000002               #-649  (1)  #-688  (1)
HP$V_WASALL                     =00000001               #-1930  (4)  #-1935  (4)  #-2071  (4)
HP$W_SIZE                        00000008               #-482  (1)  #-689  (1)
HP$W_VECTOR                      00000024               #-565  (1)
IARG$_BASE                      00000008    1048  (3)  #-1068  (3)
IARG$_PC                        00000004    1048  (3)  #-1067  (3)
IARG$_PROMPT                    00000010    1048  (3)    1066  (3)
IARG$_TEXT                      0000000C    1048  (3)    1063  (3)
IF_ALPHA                        000004C5-R   813  (3)  #-886  (3)  #-894  (3)
INI$PTABLE                      00000EC2-R  2180  (6)
```

```
INS$PTABLE                      00001064-R  2492  (6)      714  (2)
INTERPRET                       0000063B-R  1071  (3)  #-1094  (3)   #-1102  (3)   #-1109  (3)   #-1116  (3)
                                            #-1120  (3)   #-1129  (3)   #-1158  (3)   #-1175  (3)
                                            #-1183  (3)   #-1222  (3)   #-1265  (3)
KB_CHECK                        00000000-XR               179  (1)    332  (1)
L$A_DEVP                        0000021C                #-2428  (6)
L$L_HEADLENGTH                  00000200                #-2426  (6)
LOC$ADAPTER                     00000F66-R  2292  (6)  #-1622  (4)   #-1737  (4)   #-1823  (4)   #-1996  (4)
LOC$PTABLE                      00000FBB-R  2358  (6)  #-1741  (4)   #-1865  (4)   #-2032  (4)   #-2305  (6)
                                                         2497  (6)    551  (1)
LOC$PTDESC                      0000100A-R  2422  (6)    2116  (6)    463  (1)
LOCAL                           FFFFFF70    1054  (3)    1060  (3)
LOCAL_BLOCK                     FFFFFFE7     427  (1)     431  (1)
MAX_UNIT                        FFFFFFFD     427  (1)  #-681  (1)   #-757  (3)   #-761  (3)
NAME_FLAGS                      FFFFFFFF     427  (1)  #-615  (1)     679  (1)   #-752  (3)   #-760  (3)
                                                      #-767  (3)   #-776  (3)     788  (3)     792  (3)
                                                      #-803  (3)   #-854  (3)
CLDPC                           FFFFFFFC    1054  (3)  #-1072  (3)   #-1270  (3)
PARENT_CONTROLLER               FFFFFFFE     427  (1)  #-636  (1)   #-754  (3)   #-790  (3)   #-800  (3)
                                                      #-898  (3)
PARENT_PTABLE                   FFFFFFF5     427  (1)  #-545  (1)   #-552  (1)   #-773  (3)
PARSE_DEVICE                    000004DA-R   852  (3)  #-674  (1)
PART_LIST                       000002A9-R   233  (1)    234  (1)     235  (1)     236  (1)     237  (1)
                                                         238  (1)   #-613  (1)     614  (1)   #-622  (1)
                                                         623  (1)
PD$_ADD                         =0000008A    152  (1)
PD$_CASE                        =0000008C    152  (1)
PD$_COMPLEMENT                  =00000089    152  (1)
PD$_DECIMAL                     =00000082    152  (1)  #-1426  (4)
PD$_END                         =00000081    152  (1)
PD$_FETCH                       =00000087    152  (1)
PD$_HEXADECIMAL                 =00000084    152  (1)
PD$_LITERAL                     =00000086    152  (1)
PD$_LOGICAL                     =0000008B    152  (1)
PD$_NAME                        =0000008D    152  (1)  #-765  (3)
PD$_OCTAL                       =00000083    152  (1)
PD$_START                       =00000080    152  (1)  #-1069  (3)   #-1087  (3)   #-1341  (3)   #-1358  (3)
                                                      #-2452  (6)   #-763  (3)
PD$_STORE                       =00000088    152  (1)
PD$_STRING                      =00000085    152  (1)
PD_ADD                           00000699-R 1122  (3)    1087  (3)
PD_CASE                          00000728-R 1168  (3)    1087  (3)
PD_COMPLEMENT                    00000694-R 1118  (3)    1087  (3)
PD_DECIMAL                       00000754-R 1185  (3)    1087  (3)
PD_FETCH                         00000672-R 1104  (3)    1087  (3)
PD_FINISH                        00000669-R 1096  (3)    1087  (3)
PD_HEXADECIMAL                   000007B5-R 1193  (3)    1087  (3)
PD_LITERAL                       0000066D-R 1100  (3)    1087  (3)
PD_LOGICAL                       00000683-R 1131  (3)    1087  (3)
PD_NAME                          00000748-R 1180  (3)    1087  (3)
PD_NUMERIC                       000007E9-R 1197  (3)  #-1187  (3)   #-1191  (3)   #-1195  (3)
PD_OCTAL                         00000785-R 1189  (3)    1087  (3)
PD_START                         00000664-R 1092  (3)    1087  (3)
PD_STORE                         00000683-R 1111  (3)    1087  (3)
PD_STRING                        00000851-R 1236  (3)    1087  (3)
PT$EXTRACT                       000008E0-R 1336  (3)    2127  (6)
PT$INTERPRET                     00000612-R 1058  (3)     709  (1)
```

```
PTD$M_CONTROLLER                =00000002      151  (1)       151  (1)    #-950  (3)
PTD$M_DEVICE                    =0000C003      151  (1)
PTD$M_ENDDEVICE                 =0000001B      151  (1)
PTD$M_INHERIT                   =00000018      151  (1)       151  (1)    #-775  (3)
PTD$M_INHERITED                 =00000008      151  (1)
PTD$M_INHERIT_CON               =00000010      151  (1)       151  (1)
PTD$M_INHERIT_PRE               =00000008      151  (1)       151  (1)
PTD$M_NAME                      =00000004      151  (1)    #-802  (3)
PTD$M_UNIT                      =00000001      151  (1)       151  (1)    #-759  (3)    #-801  (3)    #-950  (3)
PTD$V_CONTROLLER                =00000001      151  (1)    #-630  (1)    #-884  (3)
PTD$V_INHERIT_CON               =00000004      151  (1)    #-634  (1)    #-787  (3)
PTD$V_INHERIT_PRE               =00000003      151  (1)    #-791  (3)
PTD$V_NAME                      =00000002      151  (1)    #-617  (1)    #-856  (3)
PTD$V_UNIT                      =00000000      151  (1)    #-627  (1)    #-678  (1)    #-907  (3)    #-911  (3)
PTDESC                           FFFFFFF9      427  (1)    #-468  (1)    #-755  (3)
PT_LOCATE                        00001035-R   2441  (6)    #-2430 (6)    #-2434 (6)
Q_ADAPTER                        00000002-R    261  (1)    #-1744 (4)    #-1870 (4)    #-2037  (4)    #-2222  (6)
                                                          #-2296 (6)    #-2310 (6)
Q_HUB                            0000029E-R    230  (1)       2134 (6)
SAVABS...                       =0000000C      1564 (4)
SCAN$ALPHA                       00000000-XR                  1136 (3)       180 (1)       782  (3)       978  (3)
SCAN$ALPHANUM                    0000C000-XR                   174 (1)       987 (3)
SCAN$DECIMAL                     00000000-XR                   180 (1)       905 (3)
SCAN$DEVICE                      00000000-XR                   174 (1)       506 (1)
SCAN$NUMERIC                     00000000-XR                  1208 (3)       174 (1)
SCAN$SPACES                      00000000-XR                  1519 (4)       175 (1)
SCAN$SYMBOL                      00000000-XR                   175 (1)       458 (1)
SCAN_ALPHA                       000005FA-R    976  (3)    #-936  (3)
SCAN_ALPHANUM                    00000606-R    985  (3)    #-946  (3)
SCRIPT$FLUSH                     00000000-XR                   172 (1)      1735 (4)      1818  (4)      1991  (4)
                                                           297 (1)
SET_ABORT                        000000FD-R    360  (1)       296 (1)
SHOWBUF_SIZ                     =00000100      2108 (6)      2110 (6)    #-2112 (6)    #-2142  (6)
SHOW_DEV                         00000E39-R    2109 (6)      1643 (4)      1696 (4)
SIZ...                          =00000001      164  (1)       151 (1)       161  (1)       164  (1)
SKIP                             0000093D-R    1384 (3)    #-1369 (3)    #-1374 (3)
SS$_BADPARAM                    =00000014                    #-1089 (3)    #-1165 (3)    #-1231  (3)    #-1288  (3)
                                                           #-1360 (3)
SS$_DEVALLOC                    =00000840                    #-1946 (4)
SS$_DEVALRALLOC                 =00000641                    #-1933 (4)
SS$_INSFARG                     =00000114                    #-1528 (4)
SS$_NORMAL                      =00000001                    #-1097 (3)    #-1408 (3)    #-1522  (4)    #-1922  (4)
                                                           #-2186 (6)    #-2530 (6)
SS$_NOSUCHDEV                   =00000908                    #-1943 (4)
ST_DECIMAL                       0000099B-R    1428 (4)      1426 (4)
ST_HEXADECIMAL                   000009BD-R    1436 (4)      1426 (4)
ST_LOGICAL                       000009F9-R    1453 (4)      1426 (4)
ST_OCTAL                         000009AB-R    1432 (4)      1426 (4)
ST_STRING                        000009CE-R    1440 (4)      1426 (4)
SYS$ALLOC                        00000000-XR                   176 (1)      1932 (4)
SYS$DALLOC                       00000000-XR                   176 (1)      2073 (4)
SYS$FAO                          00000000-XR                  1473 (4)       176 (1)
SYS$FAOL                         00000000-XR                  1572 (4)
T_AMBIG_ADAP                     00000161-R    210  (1)      2252 (6)
T_AMBIG_DEV                      0000018E-R    212  (1)      2254 (6)
T_BAD_NAME                       0000002E-R    191  (1)       659 (1)
T_DEATTACH                       000001C5-R    214  (1)      1777 (4)
```

T_DEVALLOC 000001E1-R 216 (1) 1945 (4)
T_INVALID 00000007-R 189 (1) 448 (1) 495 (1)
T_LINK 000000F6-R 202 (1) 494 (1) 501 (1) 537 (1)
T_MAIN_BUS 000000A7-R 197 (1) 539 (1)
T_NAME 000000EA-R 201 (1) 572 (1) 655 (1) 669 (1)
T_NO_SUCH_ADAPTER 00000124-R 206 (1) 2312 (6)
T_NO_SUCH_DEV 0000010E-R 204 (1) 1881 (4) 1942 (4) 2048 (4)
T_NO_SUCH_ON_ADAPTER 0000013B-R 208 (1) 1748 (4) 1874 (4) 2041 (4)
T_PART_OF 0000006B-R 193 (1) 654 (1)
T_PNF 00000227-R 222 (1) 321 (1)
T_PROMPT 000000DD-R 199 (1) 1526 (4)
T_SHOULDBE 00000269-R 224 (1) 1162 (3) 1227 (3) 1285 (3)
T_SHOWDEV 00000202-R 218 (1) 2137 (6)
T_SHOWDEVB 0000021C-R 220 (1) 2146 (6)
T_SKIP 00000254-R 223 (1) 325 (1)
T_STRING 00000281-R 228 (1) 1163 (3) 1286 (3)
T_TYPE 00000102-R 203 (1) 447 (1) 454 (1)
T_UNIT_EXCESS 00000075-R 195 (1) 573 (1)
T_YESNO 00000273-R 226 (1) 1161 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ALLOC_S	1	1932 (4)	1932 (4)
$D1_PRINT_S	1	1777 (4)	1777 (4)
$DALLOC_S	1	2073 (4)	2073 (4)
$DEF	1	164 (1)	
$DEFINI	1	153 (1)	153 (1) 154 (1) 155 (1) 157 (1) 158 (1) 159 (1)
$DS_CNTRLC_S	1	296 (1)	296 (1) 348 (1)
$DS_DSADEF	1	155 (1)	155 (1) 162 (1)
$DS_DSDEF	2	156 (1)	156 (1)
$DS_HDRDEF	2	153 (1)	153 (1)
$DS_HPODEF	2	154 (1)	154 (1)
$DS_PDDEF	1	152 (1)	152 (1)
$DS_PTDDEF	1	151 (1)	151 (1)
$DS_TYPEDEF	4	163 (1)	163 (1)
$EQU	1	164 (1)	151 (1) 152 (1) 156 (1) 163 (1)
$EQULS1	1	163 (1)	152 (1) 156 (1) 163 (1)
$EQULST	1	152 (1)	152 (1) 156 (1) 163 (1)
$FAOL_S	1	1571 (4)	1571 (4)
$GBLINI	2	151 (1)	151 (1) 152 (1) 156 (1) 161 (1) 163 (1) 164 (1)
$IPLDEF	1	159 (1)	159 (1)
$OFFSET	2	415 (1)	1044 (3) 1050 (3) 1331 (3) 1562 (4) 415 (1) 419 (1)
$OFFST1	1	417 (1)	1048 (3) 1054 (3) 1334 (3) 1564 (4) 417 (1) 427 (1)
$PRDEF	4	157 (1)	157 (1)
$PRINT	2	1774 (4)	1774 (4)
$PUSHADR	1	296 (1)	1572 (4) 1932 (4) 2073 (4) 2524 (6) 296 (1) 348 (1) 352 (1)
$PUSHTWO	1	1932 (4)	1932 (4)
$SSDEF	21	158 (1)	158 (1)
$VIELD	1	151 (1)	151 (1) 161 (1) 164 (1)
$VIELD1	1	164 (1)	151 (1) 161 (1) 164 (1)
BR_IF_INHIBIT_ALLOCATE	1	1925 (4)	1925 (4)
CASE	1	1073 (3)	1073 (3) 1344 (3) 1421 (4)
CLIDEF	3	160 (1)	160 (1)
CRDDEF	1	164 (1)	164 (1)
DSFDEF	3	161 (1)	161 (1)
ERRSUP_S	1	352 (1)	2524 (6) 352 (1)
MODNAM	1	188 (1)	188 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.10	00:00:00.26
Command processing	139	00:00:00.72	00:00:01.51
Pass 1	1409	00:00:21.99	00:00:26.52

Symbol table sort	16	00:00:01.80	00:00:01.94
Pass 2	849	00:00:06.92	00:00:10.75
Symbol table output	2	00:00:00.31	00:00:00.82
Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	149	00:00:02.00	00:00:03.38
Assembler run totals	2605	00:00:33.88	00:00:45.25

The working set limit was 1000 pages.
130188 bytes (255 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 989 non-local and 183 local symbols.
2533 source lines were read in Pass 1, producing 0 object records in Pass 2.
107 pages of virtual memory were used to define 37 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	10
DRB1:[DS.WORK]DS.MLB;218	7
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	32

1066 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ATTACH/UPDA=(ATTACH.UPD,ATTACH.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA

Table of contents

(2) 118 Declarations
(3) 181 DRIVER FIXED DATA AREA
(4) 212 BOO$QIO - BOOTSTRAP QIO ROUTINE
(5) 431 BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
(6) 489.3 BOO$PURDPR - Purge UBA Buffered Datapath
(8) 535.54 BOO$SELECT - Select boot driver

```
                        0000     .1          .TITLE  BOOTDRIVR       *** BOOTDRIVR non-interrupt I/O
                        0000     .2          .IDENT  'V02-017'
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000     .1 ;*  COPYRIGHT (c) 1979, 1980, 1983                                           *
-1                      0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                        C000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ;++
                        0000     27 ;
                        0000     28 ;  FACILITY:
                        0000     29 ;
                        0000     30 ;       Minimal bootstrap driver for all VMS system disks.
                        0000     31 ;
                        0000     32 ;  ENVIRONMENT:
                        0000     33 ;
                        0000     34 ;       Runs at IPL 31, kernel mode, memory management may be on or off,
                        0000     35 ;       IS=1 (running on interrupt stack), code must be PIC.
                        0000     36 ;
                        0000     37 ;  ABSTRACT:
                        0000     38 ;
                        0000     39 ;       This module contains a routine called BOO$QIO that handles I/O
                        0000     40 ;       transfers to and from the VMS system disks.
                        0000     41 ;
                        0000     42 ;  AUTHOR:
                        0000     43 ;
                        0000     44 ;       The VMS group
                        0000     .1 ;       DS02    Marion Baggett, 4-March-1983 Version 6.11
                        0000     .2 ;               Added support for HSCCI booting.
                        0000     .3 ;
                        0000     .4 ;       DS01    Roger Riggs, 24-oct-1980, version 6.1
                        0000     .5 ;               Added changes to version 4 VMB/BOOTDRIVR to make it usable
                        0000     .6 ;               as bootdrivers for the Diagnostic Supervisor.
                        0000     .7 ;               Note that some changes are here an others are in
                        0000     .8 ;               BOOTDRIVR.ENH, since there are conflicts between VMS
                        0000     .9 ;               changes and DS changes.
-2                      0000     47 ;
                        0000     .1 ;       0217    Bob Bergazzi    27-May-1983
                        0000     .2 ;               Added another dispatch for CPU XXX in the CPUDISP MACRO.
                        0000     .3 ;               This edit was put in by the supervisor group in the .ENH
```

```
0000     .4 ;                file because the edit applies to a change in the .ENH file.
0000     .5 ;                Eventually a new BOOTDRIVR module may be taken from VMS
0000     .6 ;                which has the correct code in it.
0000     .7 ;
0000     .8 ;        0216    CAS0001         C.A. Samuelson          30-Apr-1980
0000     .9 ;                Add Purge of Buffered Datapath for UBA boot drivers
0000    .10 ;
0000     48 ;        0215    TMH0001         Tim Halvorsen           10-Mar-1980
0000     49 ;                Mask off upper bits of PTE when constructing map register.
0000     50 ;
0000     51 ;        0214    KDM0092         Kathleen D. Morse       23-Jan-1980     16:40
0000     52 ;                Change VMB version number from 2 to 3.
0000     53 ;
0000     54 ;        00213   SRB0005         Steve Beckhardt         27-Nov-1979     16:10
0000     55 ;                Added code to support passing driver name from
0000     56 ;                boot driver to SYSBOOT.  Also added version # vector.
0000     57 ;        0212    SRB0004         Steve Beckhardt         31-Oct-1979     10:45
0000     58 ;                Moved all drivers into separate modules.  Added code
0000     59 ;                for connecting only the necessary driver.
0000     60 ;        0211    SRB0003         Steve Beckhardt         25-Oct-1979     16:40
0000     61 ;                Added console TU58 driver.
0000     62 ;        0210    CHF0001         Charlie Franks          23-Sep-1979     15:50
0000     63 ;                Added code and definitions to support RM80 bootable
0000     64 ;                system, includes: SSE and SSEI bit definitions, additions
0000     65 ;                to the MBAGEOM table and the MBA_DISK_DRUVER routine, and
0000     66 ;                the SSERROR skip sector routine.
0000     67 ;                Corrected byte count remaining update after encountering
0000     68 ;                a transfer error in MBA_DISK_DRIVER (convert MBA$L_BCR from
0000     69 ;                word to longword).
0000     70 ;        0209    SRB0002         Steve Beckhardt         13-Sep-1979     16:50
0000     71 ;                Added symbol EXE$GB_CPUTYPE (removed from module VMB)
0000     72 ;                so that it gets loaded into non-paged pool with the
0000     73 ;                boot driver.
0000     74 ;        0208    SRB0001         Steve Beckhardt         27-Aug-1979     10:30
0000     75 ;                Added boot device definitions ($BTDDEF).  Fixed CPUDISP
0000     76 ;                macro.  Added support for booting from console block
0000     77 ;                storage device.  Added 11/780 console floppy driver.
0000     78 ;                Changed boot value of R0 when booting from RL01/2 from
0000     79 ;                0 to 2.
0000     80 ;        0207    CHP0006         C. Peters               23-Jul-1979     16:16
0000     81 ;                Correct calculation of number of map registers needed.
0000     82 ;                Always invalidate one past the number required.
0000     83 ;        0206    CHP0005         C. Peters               20-Jul-1979     12:14
0000     84 ;                Add geometry of RM05 to MBAGEOM table.
0000     85 ;        0205    CHP0004         C. Peters               06-Jul-1979     14:06
0000     86 ;                Recode RK06/7 drive initialization code to correct bug
0000     87 ;                in RK07 handling.
0000     88 ;        0204    CHP0003         C. Peters               24-May-1979     14:24
0000     89 ;                Only invalidate the last UBA map register if the
0000     90 ;                transfer was page-aligned. Remove all code relating
0000     91 ;                to COMET-specific UBA map register format. All UBA
0000     92 ;                map registers are now alike. Delete definition of
0000     93 ;                storage for CPU_TYPE. Use instead a global symbol
0000     94 ;                EXE$GB_CPUTYPE initialized by VMB to hold the CPU
0000     95 ;                identification code.
0000     96 ;        0203    CHP0002         C. Peters               11-May-1979     13:20
0000     97 ;                Add code to invalidate the last UBA map register to stop
```

```
                                0000     98 ;                  a wild transfer. This code is CPU-specific.
                                0000     99 ;          0202    CHP0001         C. Peters                23-Mar-1979     8:48
                                0000    100 ;                  Fill out module header. Add CPUDISP macro. Add local
                                0000    101 ;                  symbol CPU_TYPE for CPUDISP's use. Add many comments.
                                0000    102 ;                  Extract processor type from PR$_SID and store in local
                                0000    103 ;                  CPU_TYPE. Find adapter type from CONFREG, and store in
                                0000    104 ;                  general register. Initialize map registers in a CPU-
                                0000    105 ;                  and adapter-dependent way. Determine which driver to
                                0000    106 ;                  use based on adapter and device type. Add routine
                                0000    107 ;                  header comments to MBA and RK06/7 driver code. Derive
                                0000    108 ;                  device CSR address in RK06/7 driver from PR$_MAPEN
                                0000    109 ;                  value used as index into RPB. Add routine header
                                0000    110 ;                  comments to ECC routine. Replace .PAGEs with form feeds.
                                0000    111 ;                  Add code and definitions to support RL02 bootable
                                0000    112 ;                  system: includes $RLDEF, a test for the RL02 device
                                0000    113 ;                  type, and the minimal RL02 driver code (written by C.
                                0000    114 ;                  Franks).
                                0000    115 ;
                                0000    116 ;--
```

```
                        0000   118          .SBTTL  Declarations
                        0000   119
                        0000   120 ;
                        0000   121 ; MACRO LIBRARY CALLS
                        0000   122 ;
                        0000   123
                        0000   124          $BTDDEF                         ; Define boot device types
                        0000   125          $IODEF                          ; DEFINE I/O FUNCTION CODES
                        0000   126          $MBADEF                         ; DEFINE MASSBUS ADAPTER REGISTERS
                        0000   127          $NDTDEF                         ; NEXUS device types
                        0000   128          $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                        0000   129          $PTEDEF                         ; DEFINE PAGE TABLE ENTRY FIELDS
                        0000   130          $RPBDEF                         ; DEFINE RESTART PARAMETER BLOCK
                        0000   131          $SSDEF                          ; DEFINE STATUS CODES
                        0000   132          $UBADEF                         ; UNIBUS ADAPTER REGISTER DEFINITIONS
                        0000   133          $UBIDEF                         ; 11/750 UNIBUS adapter regs.
                        0000   134          $VADEF                          ; DEFINE VIRTUAL ADDRESS FIELDS
                        0000   135
                        0000   136 ;
                        0000   137 ; MACROS
                        0000   138 ;
                        0000   139
                        0000   140 ;
                        0000   141 ; CPU type dispatch macro:
                        0000   142 ;
                        0000   143 ;       The addresses in the address list are:
                        0000   144 ;               address of code for CPU type=1 (11/780)
                        0000   145 ;               address of code for CPU type=2 (11/750)
                        0000   146 ;               address of code for CPU type=3 (?)
                        0000   147 ;
                        0000   148 ;       CPUDISP is invoked to handle CPU differences in line, e.g.,
                        0000   149 ;
                        0000   150 ;               CPUDISP <10$,20$>       ;*DISPATCH ON CPU TYPE*
                        0000   151 ;       10$:    <11/780 specific code>
                        0000   152 ;       20$:    <11/750 specific code>
                        0000   153 ;                                       ;*END OF CPU-DEPENDENT CODE*
                        0000   154 ;
                        0000   155 ; When the next CPU is added, all occurrences of CPUDISP must be
                        0000   156 ; expanded to handle three CPU specific paths.
                        0000   157 ;
                        0000   158
                        0000   159          .MACRO  CPUDISP,ADDRLIST
                        0000    .1          CASE    W^EXE$GB_CPUTYPE,<ADDRLIST>,LIMIT=#PR$_SID_TYP780,TYPE=B
-1                      0000   161          .ENDM   CPUDISP
                        0000   162
                        0000   163 ;
                        0000   164 ; LOCAL SYMBOLS
                        0000   165 ;
                        0000   166
                        0000   167          $DEFINI BDT                     ; Define Boot Driver Table offsets
                        0000   168
                        0000   169 $DEF     BDT$L_CPUTYPE   .BLKW   1       ;  CPU type
                        0002   170 $DEF     BDT$L_DEVTYPE   .BLKW   1       ;  Boot R0 device type
                        0004   171 $DEF     BDT$L_ACTION    .BLKL   1       ;  Action routine
                        0008   172 $DEF     BDT$L_SIZE      .BLKL   1       ;  Driver size
                        000C   173 $DEF     BDT$L_ADDR      .BLKL   1       ;  Driver address (offset)
                        0010   174 $DEF     BDT$L_ENTRY     .BLKL   1       ;  Driver entry (offset from address)
```

```
                        0014   175 $DEF    BDT$L_DRIVRNAME .BLKL   1       ;  Driver name (offset from address)
                        0018   176
                00000018  0018   177 BDT$K_LENGTH=.                          ;  Length of entry
                        0018   178
                        0018   179         $DEFEND BDT                       ; End of Boot Driver Table definitions
```

```
                                0000   181          .SBTTL  DRIVER FIXED DATA AREA
                                0000   182
                                0000   183 ;
                                0000   184 ;       FIXED DATA CELLS FOR BOOTSTRAP DRIVER
                                0000   185 ;
                                0000   186
                           00000000    187          .PSECT  BOOTDRIVR_1
                                0000   188
                                0000   189 BOO$AL_VECTOR::                          ; VECTOR TO BOOT DRIVER ENTRY POINTS
                   00000016'    0000   190          .LONG   BOO$QIO-BOO$AL_VECTOR   ; OFFSET TO BOOTSTRAP QIO ROUTINE
                   000000F1'    0004   191          .LONG   BOO$MAP-BOO$AL_VECTOR   ; OFFSET TO MAPPING ROUTINE
                   00000000'    0008   192          .LONG   BOO$SELECT-BOO$AL_VECTOR; OFFSET TO BOOTSTRAP I/O DRIVER
                                000C   193                                          ; INITIALLY SET TO ROUTINE WHICH
                                000C   194                                          ; SELECTS DRIVER
                   00000000     000C   195          .LONG   0                       ; OFFSET TO SYSTEM DISK DRIVER NAME
                                0010   196                                          ; (ASCIC STRING). SET UP BY BOOT DRIVER.
                                0010   197 ;
                                0010   198 ; The next two words are the version number and the version number check fields.
                                0010   199 ; (The second word is the ones complement of the first word.)  The version
                                0010   200 ; number should be incremented whenever the interface between VMB and the
                                0010   201 ; rest of the system changes.  Release 1.0 VMB did not contain these fields.
                                0010   202 ;
                                0010   203 ;       Version 2 - Boot driver passes system disk driver name to SYSBOOT
                                0010   204 ;       Version 3 - VMB build memory description vector into RPB
                                0010    .1 ;       Version 4 - VMB BOOTDRIVR purges UBA buffered datapath, all drivers
                                0010    .2 ;                   return to BOOTDRIVR with success/failure status
                                0010    .3 ;
                       FFFB 0004    0010    .4          .WORD   4,^C<4>             ; VERSION # AND VERSION # CHECK FIELD.
-3                                  0014   208 EXE$GB_CPUTYPE::                     ; Location to hold processor
                             01     0014   209          .BYTE   1                   ; identification code
                                    0015    .1 EXE$GB_ADPTYPE::                     ; Location to hold flag for
                             01     0015    .2          .BYTE   1                   ; remebering boot adapter type.
                                    0016   210
```

```
                                0016   212          .SBTTL  BOO$QIO - BOOTSTRAP QIO ROUTINE
                                0016   213
                                0016   214 ;++
                                0016   215 ; FUNCTIONAL DESCRIPTION:
                                0016   216 ;
                                0016   217 ;       BOO$QIO PROVIDES THE DEVICE INDEPENDENT I/O INTERFACE FOR BOTH
                                0016   218 ;       READING AND WRITING THE BOOTSTRAP DEVICE.
                                0016   219 ;
                                0016   220 ; CALLING SEQUENCE:
                                0016   221 ;
                                0016   222 ;       CALLG   ARGLIST,BOO$QIO
                                0016   223 ;
                                0016   224 ; INPUT PARAMETERS:
                                0016   225 ;
                                C016   226 ;       BUF(AP)  - BUFFER ADDRESS
                                0016   227 ;       SIZE(AP) - SIZE OF BUFFER IN BYTES
                                0016   228 ;       LBN(AP)  - LOGICAL BLOCK NUMBER
                                0016   229 ;       FUNC(AP) - FUNCTION CODE
                                0016   230 ;                  ACCEPTS IO$_READLBLK AND IO$_WRITELBLK
                                0016   231 ;       MODE(AP) - ADDRESS INTERPRETATION MODE
                                0016   232 ;                  0 => PHYSICAL, 1 => VIRTUAL
                                0016   233 ;       RPB(AP) - ADDRESS OF RESTART PARAMETER BLOCK
                                0016   234 ;
                                0016   235 ; OUTPUT PARAMETERS:
                                0016   236 ;
                                0016   237 ;       R0 - COMPLETION STATUS CODE
                                0016   238 ;       R1 - TOTAL BYTES TRANSFERRED
                                0016   239 ;
                                0016   240 ;--
                                0016   241
                                0016   242 ;
                                0016   243 ; Offsets from AP to input arguments:
                                0016   244 ;
                                0016   245
                      00000004  0016   246          BUF     = 4
                      00000008  0016   247          SIZE    = 8
                      0000000C  0016   248          LBN     = 12
                      00000010  0016   249          FUNC    = 16
                      00000014  0016   250          MODE    = 20
                      00000018  0016   251          RPB     = 24
                                0016   252
                                0016   253 BOO$QIO::
                          0FFC  0016   254          .WORD   ^M<R2,R3,R4,R5,R6,R7,-  ; PRESERVE REGISTERS
                                0018   255                  R8,R9,R10,R11>
                                0018   256
                                0018   257 ;
                                0018   258 ; If mapping is enabled, the processor register RP$_MAPEN contains a 1.
                                0018   259 ; Otherwise, the register contains a 0. Use this value as an index to
                                0018   260 ; choose the appropriate address of the adapter's register space.
                                0018   261 ;
                                0018   262
     59    18 AC    D0         0018   263          MOVL    RPB(AP),R9              ; GET BASE ADDRESS OF RESTART PARAMETER BLK
        51    38    DB         001C   264          MFPR    #PR$_MAPEN,R1           ; CHECK FOR MAPPING ENABLED
                                001F   265          ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
     53    5C A941  D0         001F   266          MOVL    RPB$L_ADPPHY(R9)[R1],R3 ; GET CORRECT POINTER TO CONFIG REG
                                0024   267
                                0024   268 ;
```

```
                                0024   269 ; Using the argument list as input, calculate the transfer size, number
                                0024   270 ; of map registers, starting LBN, starting VPN, and base of a page table
                                0024   271 ; to use in mapping.
                                0024   272 ;
                                0024   273
      5A   04 AC    D0  0024   274          MOVL    BUF(AP),R10             ; Get buffer address
      58   08 AC    3C  0028   275          MOVZWL  SIZE(AP),R8             ; GET TRANSFER SIZE IN BYTES
           04       12  002C   276          BNEQ    10$                     ; CONTINUE IF LEGAL SIZE
   58  01   10       9C  002E   277          ROTL    #16,#1,R8               ; ELSE FORCE TO 64K SIZE
   5A  09   00       EF  0032   278 10$:     EXTZV   #VA$V_BYTE,#VA$S_BYTE,R10,R7    ; Get byte offset into page
            57           0036
   57  03FF C847    9E  0037   279          MOVAB   ^X3FF(R8)[R7],R7        ; Calculate highest address plus
                        003D   280                                          ; an overflow page.
      57   F7 8F    78  003D   281          ASHL    #-9,R7,R7               ; Reduce to number of pages
           57           0041
                        0042   282                                          ; (= number of map registers).
      5B   0C AC    D0  0042   283          MOVL    LBN(AP),R11             ; AND BLOCK NUMBER FOR RETRY
      51   50 A9    D0  0046   284          MOVL    RPB$L_SVASPT(R9),R1     ; ASSUME SYSTEM SPACE
      03 5A   1F    E0  004A   285          BBS     #VA$V_SYSTEM,R10,20$    ; Branch if system address
         51   08    DB  004E   286          MFPR    #PR$_P0BR,R1            ; OTHERWISE GET P0 PT BASE
   5A  15   09       EF  0051   287 20$:     EXTZV   #VA$V_VPN,#VA$S_VPN,R10,R2      ; Get base VPN for transfer
            52           0055
         FFBB CF    94  0056    .1         CLRB    W^EXE$GB_ADPTYPE        ; Assume not UNIBUS adapter.
           66 A9    91  005A   288         CMPB    RPB$B_DEVTYP(R9),-      ; If booting from console block
              20        005D    .1                 #BTD$K_HSCCI            ; storage device,                       [DSO
              64    18  005E    .2         BGEQ    PUSH_RETRY              ; don't load map registers
-2                      0060   291
                        0060   292 ;
                        0060   293 ; Derive the boot device adapter's type (UNIBUS adapter or MASSBUS
                        0060   294 ; adapter) from the RPB, and save a flag indicating the adapter type
                        0060   295 ; in a register.
                        0060   296 ;
                        0060   297
-1          54   20 A9  D0  0060   299          MOVL    RPB$L_BOOTR1(R9),R4     ; Get adapter's SBI slot number.
      54   0090 C944    9A  0064   300          MOVZBL  RPB$B_CONFREG(R9)[R4],R4; Get type of adapter.
             20   54    D1  006A   301          CMPL    R4,#NDT$_MB             ; If MASSBUS adapter,
                  04    13  006D   302          BEQL    INIT_MAPREGS            ; just leave as is, LBC.
           FFA2 CF    96  006F    .1         INCB    W^EXE$GB_ADPTYPE        ; Otherwise, set low bit to
-1                      0073   304                                          ; indicate UNIBUS adapter.
                        0073   305
                        0073   306 ;
                        0073   307 ; Register usage right now is as follows:
                        0073   308 ;
                        0073   309 ;       R0      - adapter type; LBS=UNIBUS; LBC=MASSBUS
                        0073   310 ;       R1      - address of page table for virtual-->physical mapping
                        0073   311 ;       R2      - base VPN for the transfer
                        0073   312 ;       R3      - address of the adapter's configuration register
                        0073   313 ;       R7      - number of map registers needed (plus one extra)
                        0073   314 ;       R8      - transfer size in bytes
                        0073   315 ;       R10     - buffer address
                        0073   316 ;       R11     - starting LBN of the transfer
                        0073   317 ;
                        0073   318 ; In an adapter-dependent fashion, initialize the required number of
                        0073   319 ; adapter map registers. First calculate the address of the starting map
                        0073   320 ; register number. Right now, map registers for all UNIBUS and MASSBUS
                        0073   321 ; adapters for all processors start at the same offset from the base of
                        0073   322 ; the adapter's register space.
```

```
                                0073   323 ;
                                0073   324 ; During map register initialization, the following registers change
                                0073   325 ; for each page mapped:
                                0073   326 ;
                                0073   327 ;       R2      - address of the next VPN to map
                                0073   328 ;       R4      - address of the next map register to load
                                0073   329 ;       R5      - PFN of the page being mapped
                                0073   330 ;
                                0073   331
                                0073   332 INIT_MAPREGS:                                  ; Initialize the map registers.
                                0073   333         ASSUME  MBA$L_MAP EQ UBA$L_MAP
                                0073   334         ASSUME  MBA$L_MAP EQ UBI$L_MAP
                      54  0800 C3  DE  0073   335         MOVAL   MBA$L_MAP(R3),R4        ; Point to base of adapter map.
                                0078   336
                                C078   337 COMPUTE_PFN:                                  ; Loop once per page.
                          55    82  9E  0078   338         MOVAB   (R2)+,R5                ; Get a virtual page number.
                        0B 14   AC  E9  007B   339         BLBC    MODE(AP),10$            ; If physical page #, branch.
                       55   6145  D0  007F   340         MOVL    (R1)[R5],R5             ; Get page table entry
                    FFE00000 8F  CA  0083   341         BICL    #^C<PTE$M_PFN>,R5       ; and extract PFN from entry
                             55         0089
                                008A   342
                                008A   343 10$:                                         ; Branch on adapter type.
                      10 FF87   CF  E9  008A    .1        BLBC    W^EXE$GB_ADPTYPE,LOAD_MBA_MAP ; Branch if MASSBUS adapter.
-1
                                008F   345
                                008F   346 ;
                                008F   347 ; This is a UNIBUS adapter.
                                008F   348 ;
                                008F   349 ; Map registers for the UNIBUS adapter look like the following:
                                008F   350 ;
                                008F   351 ;       +-----------------------------------------------+
                                008F   352 ;       ! V !   ! BO ! DP # !   !  page frame number   !
                                008F   353 ;       +-----------------------------------------------+
                                008F   354 ;
                                008F   355 ; The code sets the byte offset bit if relevant, sets the valid bit,
                                008F   356 ; sets the low bit in the 4-bit data path field to indicate that the
                                008F   357 ; first buffered data path is to be used, and loads the page frame
                                008F   358 ; number into the low bits.
                                008F   359 ;
                                008F   360
                   19   04  AC  F0  008F   361         INSV    BUF(AP),#UBA$V_MAP_BO,- ; Set UBA byte offset bit if
                        55  01      0093   362                 #1,R5                   ; necessary.
                            55  C9  0095   363         BISL3   R5,#UBA$M_MAP_VALID!-   ; Set PFN and byte offset, valid
                80200000    8F      0097
                            84      009C
                                009D   364                 <1@UBA$V_MAP_DPD>,(R4)+ ; bit, and buffered DP number
                                009D   365                                         ; into mao.
                                009D    .1
                                009D    .2 ; ****** NOTE *****  Always uses Datapath #1, IOC$PURGE depends on this!!
                                009D    .3
                            08  11  009D   366         BRB     LOAD_NEXT_MAP           ; Go load another map if needed.
                                009F   367
                                009F   368 ;
                                009F   369 ; MASSBUS adapter's map registers look like the following:
                                009F   370 ;
                                009F   371 ;       +-----------------------------------------+
                                009F   372 ;       ! V !                 ! page frame number !
                                009F   373 ;       +-----------------------------------------+
```

```
                                009F    374 ;
                                009F    375
                                009F    376 LOAD_MBA_MAP:                               ; Load map register.
                          55    C9  009F    377             BISL3   R5,#^X80000000,(R4)+    ; Set the PFN and the valid bit
                    80000000    8F  00A1
                          84        00A6
                                00A7    378                                             ; in the map register.
                                00A7    379
                                00A7    380 ;
                                00A7    381 ;*****
                                00A7    382 ;           BISL3   R5,#MBA$M_MAP_VALID,-   ; Set the PFN and the valid bit
                                00A7    383 ;                   (R4)+                   ; in the map register.
                                00A7    384 ;*****
                                00A7    385 ;
                                00A7    386
                                00A7    387 LOAD_NEXT_MAP:                              ; Is there another page to do?
                          57    D7  00A7    388             DECL    R7                      ; Decrement # of map registers.
                          CD    14  00A9    389             BGTR    COMPUTE_PFN             ; Loop to fill next map register
                                00AB    390                                             ; if byte count not exhausted.
                                00AB    391
                                00AB    392 ;
                                00AB    393 ; Invalidate the last UBA map register so that a wild transfer will stop
                                00AB    394 ; at the end of the last valid block.
                                00AB    395 ;
                                00AB    396
                       07 FF66  CF    E9  00AB     .1         BLBC    W^EXE$GB_ADPTYPE,GET_BYTEOFFSET ; Skip invalidation for MBA.
-1                  80000000    8F    CA  00B0    398         BICL    #UBA$M_MAP_VALID,-(R4)  ; Invalidate last map register.
                          74        00B6
                                00B7    399
                                00B7    400 ;
                                00B7    401 ; All map registers are set up. Set up 2 more inputs to driver code.
                                00B7    402 ; Since the loaded map registers are registers #0-n, the starting
                                00B7    403 ; address of the transfer to be loaded into a device register by the
                                00B7    404 ; device driver, is now simply the offset into the first page of the
                                00B7    405 ; transfer buffer.
                                00B7    406 ;
                                00B7    407
                                00B7    408 GET_BYTEOFFSET:
                          09    00    EF  00B7    409             EXTZV   #VA$V_BYTE,#VA$S_BYTE,- ; Get byte offset into page
                          5A    5A        00BA    410                     R10,R10                 ; in R10
                                00BC    411
                                00BC    412 ;
                                00BC    413 ; If it is a UNIBUS boot device, derive the address of the device's CSR.
                                00BC    414 ;
                                00BC    415
                                00BC    416             ASSUME  RPB$L_CSRVIR  EQ  RPB$L_CSRPHY+4
                                00BC    417
                          50    38    DB  00BC    418             MFPR    #PR$_MAPEN,R0           ; Check for mapping enabled
                    57    54  A940    D0  00BF    419             MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get address of device's CSR
                                00C4    420
                                00C4    421 PUSH_RETRY:
                                0A    DD  00C4    422             PUSHL   #10                     ; Push retry count on stack
                                00C6    423
                          55    5B    D0  00C6    424 10$:        MOVL    R11,R5                  ; Get a working copy of the block number
                       50     34  A9    D0  00C9    425             MOVL    RPB$L_IOVEC(R9),R0      ; Get address of boot vectors
                          08  B040    16  00CD    426             JSB     @8(R0)[R0]              ; Call driver thru self-relative vector
                       14 FF40  CF    E9  00D1     .1             BLBC    W^EXE$GB_ADPTYPE,100$   ; Branch if Mass Bus
```

```
                      50   DD  00D6    .2          PUSHL   R0                      ; Save driver status
                      02   BB  00D8    .3          PUSHR   #^M<R1>                 ; Save R1 from driver
                    006C   30  00DA    .4          BSBW    BOO$PURDPR              ; Purge Buffered Datapath for UBA
                      02   BA  00DD    .5          POPR    #^M<R1>                 ; Restore R1
                   05 50   E8  00DF    .6          BLBS    R0,80$                  ; Branch if success
                5E    04   C0  00E2    .7          ADDL    #4,SP                   ; Clear previous status from stack
                      06   11  00E5    .8          BRB     150$                    ; Retry
                      50 8ED0  00E7    .9 80$:     POPL    R0                      ; Get driver status back
                   03 50   E8  00EA   .10 100$:    BLBS    R0,200$                 ; Branch if success
                   D6 6E   F5  00ED   .11 150$:    SOBGTR  (SP),10$                ; Retry if count > 0
                           04  00F0   .12 200$:    RET                             ; Return with final status in R0
```

```
                                00F1   431          .SBTTL  BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
                                00F1   432
                                00F1   433 ;++
                                00F1   434 ; FUNCTIONAL DESCRIPTION:
                                00F1   435 ;       BOO$MAP IS CALLED TO INITIALIZE THE DATA BASE FOR BOO$QIO TO PERMIT
                                00F1   436 ;       IT TO FUNCTION WITH MEMORY MANAGEMENT ENABLED.  AN AREA OF SYSTEM
                                00F1   437 ;       PAGE TABLE MUST BE PROVIDED SO THAT THE CONFIGURATION REGISTERS AND
                                00F1   438 ;       UNIBUS I/O PAGE CAN BE MAPPED.
                                00F1   439 ;
                                00F1   440 ; CALLING SEQUENCE:
                                00F1   441 ;       CALLG   ARGLIST,BOO$MAP
                                00F1   442 ;
                                00F1   443 ; INPUT PARAMETERS:
                                00F1   444 ;       SVASPT(AP) - SYSTEM VIRTUAL ADDRESS OF THE SYSTEM PAGE TABLE
                      00000004  00F1   445 ;       SVASPT  =    4
                                00F1   446 ;       VABASE(AP) - BASE VIRTUAL ADDRESS OF A 24 PAGE WINDOW TO MAP
                                00F1   447 ;                    THE ADAPTER CONFIGURATION REGISTERS AND UNIBUS
                      00000008  00F1   448 ;       VABASE  =    8
                                00F1   449 ;                    I/O PAGE.
                                00F1   450 ;       RPB(AP) -    ADDRESS OF RESTART PARAMETER BLOCK (RPB) CONTAINING
                                00F1   451 ;                    BOOTSTRAP DEVICE DESCRIPTION.
                      0000000C  00F1   452 ;       RPB     =    12
                                00F1   453 ;
                                00F1   454 ; OUTPUT PARAMETERS:
                                00F1   455 ;       NONE
                                00F1   456 ;
                                00F1   457 ;--
                                00F1   458
                          00FC  00F1   459 BOO$MAP::      .WORD   ^M<R2,R3,R4,R5,R6,R7>   ;
    57    0C AC      D0      00F3   460          MOVL    RPB(AP),R7              ; GET BASE ADDRESS FOR RPB
    52    04 AC      D0      00F7   461          MOVL    SVASPT(AP),R2           ; GET BASE OF SPT
 50 A7      52      D0      00FB   462          MOVL    R2,RPB$L_SVASPT(R7)     ; AND SAVE IN DATA BASE
    53    08 AC      D0      00FF   463          MOVL    VABASE(AP),R3           ; GET VIRTUAL ADDRESS OF WINDOW
 60 A7      53      D0      0103   464          MOVL    R3,RPB$L_ADPVIR(R7)     ; SET AS ADAPTER VIRTUAL ADDRESS
       15    09      EF      0107   465          EXTZV   #VA$V_VPN,#VA$S_VPN,RPB$L_ADPPHY(R7),R4 ; GET BASE PFN
    54    5C A7              010A
       55    08      D0      010D   466          MOVL    #8,R5                   ; SET TO MAP 8 PAGES
 53    15    09      EF      0110   467          EXTZV   #VA$V_VPN,#VA$S_VPN,R3,R0        ; GET BASE VIRTUAL PAGE
             50              0114
    51    6240      DE      0115   468          MOVAL   (R2)[R0],R1             ; COMPUTE WORKING SPT POINTER
             20      10      0119   469          BSBB    FILLSPT                 ; FILL SPT TO MAP CONFIGURATION REGS
       55    10      D0      011B   470          MOVL    #16,R5                  ; SET FOR 16 PAGES
  00001FFF  8F      CB      011E   471          BICL3   #^X1FFF,RPB$L_CSRPHY(R7),R4     ; GET PHY ADDR OF I/O PAGE BASE
    54    54 A7              0124
 54    54    17      9C      0127   472          ROTL    #<32-9>,R4,R4           ; AND CONVERT TO PAGE NUMBER
             0E      10      0128   473          BSBB    FILLSPT                 ; STORE PTES INTO SPT
    50    54 A7      3C      012D   474          MOVZWL  RPB$L_CSRPHY(R7),R0     ; GET I/O PAGE OFFSET
FFFF3000  E043      9E      0131   475          MOVAB   <^X1000-^XE000>(R0)[R3],RPB$L_CSRVIR(R7) ; SET VIRTUAL CSR ADDR
          58 A7              0138
             04      013A   476          RET                             ;
                     013B   477
                     013B   478 ;++
                     013B   479 ; FILLSPT
                     013B   480 ;
                     013B   481 ; INPUTS:
                     013B   482 ;       R1 - POINTER TO CURRENT SPT ENTRY (UPDATED)
                     013B   483 ;       R4 - PFN (UPDATED)
```

```
                                013B   484 ;       R5 - COUNT OF PAGES TO FILL (UPDATED)
                                013B   485 ;
                                013B   486 FILLSPT:                                             ;
                    90000000 8F   C9  013B   487         BISL3   #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+ ; STORE A PTE
                          81   54       0141
                               54   D6  0143   488         INCL    R4                          ; ADVANCE TO NEXT PFN
                             F3 55   F5  0145   489         SOBGTR  R5,FILLSPT                  ; STORE THEM ALL
                                 05   0148    .1         RSB
```

```
                                0149    .3          .SBTTL  B00$PURDPR - Purge UBA Buffered Datapath
-20                             0149    510
                                0149    511 ;++
                                0149    512 ; FUNCTIONAL DESCRIPTION:
                                0149    513 ;
                                0149     .1 ;       This routine is called by BOOTDRIVR at the end of each boot device
                                0149     .2 ;       transfer if the boot device is on the Unibus.  It purges the buffered
                                0149     .3 ;       datapath and/or performs other Unibus adapter specific end-action.
                                0149     .4 ;
                                0149     .5 ;   NOTE: This routine contains processor specific code.
-6                              0149    520 ;
                                0149    521 ; CALLING SEQUENCE:
                                0149    522 ;
                                0149     .1 ;       JSB     B00$PURDPR
-2                              0149    525 ;
                                0149    526 ; INPUT PARAMETERS:
                                0149    527 ;
                                0149     .1 ;       R3 - Address of UBA adapter configuration register
                                0149     .2 ;       EXE$GB_CPUTYPE - Index specifying what CPU we are executing on
                                0149     .3 ;       ** Assumes all drivers use DATAPATH 1 **
-1                              0149    529 ;
                                0149    530 ; OUTPUT PARAMETERS:
                                0149    531 ;
                                0149     .1 ;       R0 - LBS -> Success
                                0149     .2 ;            LBC -> Failure
                                0149     .3 ;
                                0149     .4 ;       R1,R2,R4 - Destroyed
                                0149     .5 ;       All other registers preserved
-1                              0149    533 ;
                                0149    534 ;--
                                0149    535
                                0149     .1 B00$PURDPR:
                                0149     .2
      50    01    3C            0149     .3          MOVZWL  #SS$_NORMAL,R0          ; Assume success
                                014C     .4          CPUDISP <100$,200$,300$,100$>   ; Dispatch on EXE$GB_CPUTYPE
                                015A     .5
    52    44 A3    DE            015A     .6 100$:    MOVAL   UBA$L_DPR+4(R3),R2      ; CPU type 11/780, Datapth Register
62    01    1F    78            015E     .7          ASHL    #UBA$V_DPR_BNE,#1,(R2)  ; Purge datapath
      51    62    D0            0162     .8          MOVL    (R2),R1                 ; Get Datapth register contents
  09  51    1E    E1            0165     .9          BBC     #UBA$V_DPR_XMTER,R1,170$ ; Branch if no error
62    01    1E    78            0169    .10          ASHL    #UBA$V_DPR_XMTER,#1,(R2) ; Clear error in datapath
50    01F4  8F    3C            016D    .11 150$:    MOVZWL  #SS$_PARITY,R0          ; Set failure status
                  05            0172    .12 170$:    RSB                             ; Return to caller
                                0173    .13
    52    04 A3    DE            0173    .14 200$:    MOVAL   UBI$L_DPR+4(R3),R2      ; CPU type 11/750, Datapath Register
62    01    00    78            0177    .15          ASHL    #UBI$V_DPR_PUR,#1,(R2)  ; Purge Datapath
      54    0A    D0            017B    .16          MOVL    #UBI$C_PURCNT,R4        ; Get max # of tries for
                                017E    .17                                           ; purge done test
      51    62    D0            017E    .18 230$:    MOVL    (R2),R1                 ; Get datapath register contents
  05  51    00    E1            0181    .19          BBC     #UBI$V_DPR_PUR,R1,250$  ; Branch if purge done
        F6  54    F5            0185    .20          SOBGTR  R4,230$                 ; Branch if more tries allowed
                  04    11      0188    .21          BRB     270$                    ; Return failure status
  E4  51    1F    E1            018A    .22 250$:    BBC     #UBI$V_DPR_ERROR,R1,170$ ; Branch if no purge error
      62    00    D2            018E    .23 270$:    MCOML   #0,(R2)                 ; Clear datapath error(s)
                  DA    11      0191    .24          BRB     150$                    ; Return with failure status
                                0193    .25
    51    10 A3    D0            0193    .26 300$:    MOVL    UBI$L_SR(R3),R1         ; Get Unibus Error Summary Register
```

```
                          0197    .27                                           ; Nebula
        8001C000 8F   D3  0197    .28          BITL     #<UBI$M_SR_UWE!-        ; Any UB errors? (write error,
                 51       019D
                          019E    .29                   UBI$M_SR_MRPE!-         ;       map parity error,
                          019E    .30                   UBI$M_SR_NXM!-          ;       non-existent memory,
                          019E    .31                   UBI$M_SR_UCE>,R1        ; or uncorrected read error.)
                 D2   13  019E    .32          BEQL     170$                    ; Branch if no errors
                          01A0    .33 ; ****** QUESTION - Is there anything to do to clear the error status?
                 CB   11  01A0    .34          BRB      150$                    ; Return failure status
                          01A2    .35
```

```
000001A2  01A2    .37 BOO$QIOSIZ=.-BOO$AL_VECTOR              ; Size of boot QIO routine
          01A2    .38
000001A2  01A2    .39 BOO$DRIVER==.                           ; Start of boot driver (after
          01A2    .40                                         ; it's been moved)
          01A2    .41                                         ; NOTE:  Boot drivers must be in
          01A2    .42                                         ; psect BOOTDRIVR_2
          01A2    .43
          00000000  .44         .PSECT  BOOTDRIVR_3
              0000  .45
00000000  0000    .46 BOO$DRIVER_TBL=.                        ; Boot driver table
              0000  .47
          00000000  .48         .PSECT  BOOTDRIVR_5
              0000  .49
00000000  0000    .50         .LONG   0                       ; End of boot driver table
              0004  .51
          00000000  .52         .PSECT  BOOTDRIVR_6
```

```
                                0000    .54          .SBTTL  BOO$SELECT - Select boot driver
                                0000    .55
                                0000    .56 ;++
                                0000    .57 ; FUNCTIONAL DESCRIPTION:
                                0000    .58 ;
                                0000    .59 ;       This routine is called the first time BOO$QIO calls a driver.
                                0000    .60 ;       It searches the boot driver table to locate the proper driver.
                                0000    .61 ;       This driver is then moved so that it is adjacent to BOO$QIO
                                0000    .62 ;       and the correct linkage is made in BOO$AL_VECTOR.
                                0000    .63 ;       RPB$L_IOVECSZ is also stored with the size of BOO$QIO plus
                                0000    .64 ;       the size of the driver.  The driver is then jumped to.
                                0000    .65 ;
                                0000    .66 ; CALLING SEQUENCE:
                                0000    .67 ;
                                C000    .68 ;       JSB     BOO$SELECT      (Actually called through self-relative
                                0000    .69 ;                                vector in BOO$AL_VECTOR+8)
                                0000    .70 ;
                                0000    .71 ; INPUT PARAMETERS:
                                0000    .72 ;
                                0000    .73 ;       R9      Address of the RPB
                                0000    .74 ;
                                0000    .75 ; OUTPUT PARAMETERS:
                                0000    .76 ;
                                0000    .77 ;       None
                                0000    .78 ;
                                0000    .79 ;--
                                0000    .80
                                0000    .81 BOO$SELECT::                    ; Global so DS can re-init BOO$AL_VECTOR
-1                        3E    BB    0000   537          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save registers
                  55  0000'CF    DE    0002    .1          MOVAL   W^BOO$DRIVER_TBL,R5     ; Get address of boot driver table
                    53    66 A9    9A    0007    .2          MOVZBL  RPB$B_DEVTYP(R9),R3     ; Get value of boot device type
                  54  0014'CF    9A    000B    .3          MOVZBL  W^EXE$GB_CPUTYPE,R4     ; Get cpu type
-3                                      0010   541
                                0010    542 10$:    ;
                                0010    543         ; Determine if next driver in table is the correct one.
                                0010    544         ;
                                0010    545
                     50    65    32    0010   546          CVTWL   BDT$L_CPUTYPE(R5),R0    ; Get cpu type from table
                           53    13    0013   547          BEQL    80$                     ; End of table
                           05    19    0015   548          BLSS    20$                     ; Driver doesn't care about cpu type
                     54    50    D1    0017   549          CMPL    R0,R4                   ; Cpu types match?
                           17    12    001A   550          BNEQ    40$                     ; No, try next driver
                                001C    551
                  50    02 A5    32    001C   552 20$:    CVTWL   BDT$L_DEVTYPE(R5),R0    ; Get boot device type from table
                           05    19    0020   553          BLSS    30$                     ; Driver doesn't care about device type
                     53    50    D1    0022   554          CMPL    R0,R3                   ; Device types match?
                           0C    12    0025   555          BNEQ    40$                     ; No, try next driver
                                0027    556
                  50    04 A5    D0    0027   557 30$:    MOVL    BDT$L_ACTION(R5),R0     ; Get action routine offset from table
                           0B    13    002B   558          BEQL    60$                     ; No action routine, this is the driver
                        6540    16    002D   559          JSB     (R5)[R0]                ; Call action routine
                     05    50    E8    0030   560          BLBS    R0,60$                  ; Branch if this is the driver
                     55    18    C0    0033   561 40$:    ADDL    #BDT$K_LENGTH,R5        ; Point to next driver entry
                           D8    11    0036   562          BRB     10$                     ; Try next driver
                                0038    563
                                0038    564 60$:    ;
                                0038    565         ; Have the right driver.  R5 points to driver table entry.
```

```
                                0038   566          ; Set up size in RPB, driver vector, and then move driver.
                                0038   567          ; Finally, jump to driver.
                                0038   568
      000001A2 8F    C1        0038   569          ADDL3   #BOO$QIOSIZ,-                       ; Add boot QIO size to
    38 A9    08 A5              003E   570                  BDT$L_SIZE(R5),RPB$L_IOVECSZ(R9); driver size and store in RPB
      00000000'8F    C3        0042    .1          SUBL3   #BOO$AL_VECTOR, R5, R0  ; Offset from vector to header
           50    55            0048
        10 A5    50    C1      004A    .2          ADDL3   R0,-
           0008'CF               004E
                                0051    .3                  BDT$L_ENTRY(R5),W^BOO$AL_VECTOR+8 ; entry and store in vector
           0C A5    C0         0051    .4          ADDL2   BDT$L_ADDR(R5),W^BOO$AL_VECTOR+8 ; Plus offset from header
           0008'CF              0054
        14 A5    50    C1      0057    .5          ADDL3   R0,-                        ; Compute offset to driver
           000C'CF              005B
                                C05E    .6                  BDT$L_DRIVRNAME(R5),W^BOO$AL_VECTOR+12 ; name and store in vector
                                005E    .7 ; Do not move the driver in the supervisor
                                005E    .8 ;        MOVC3   BDT$L_SIZE(R5),-                    ; Move driver
                                005E    .9 ;                @BDT$L_ADDR(R5)[R5],W^BOO$DRIVER
-6                 3E    BA     005E   577          POPR    #^M<R1,R2,R3,R4,R5>                 ; Restore registers
           50    34 A9    D0     0060   578          MOVL    RPB$L_IOVEC(R9),R0                  ; Get address of vectors
              08 B040    17     0064   579          JMP     @8(R0)[R0]                          ; Jump to driver
                                0068   580
                                0068   581
                                0068   582 80$:     ;
                                0068   583          ; No driver in the driver table accepted this QIO.
                                0068   584          ;
                                0068   585
                          00    0068   586          HALT
                                0069   587          .END
```

```
BDT$K_LENGTH                    = 00000018     D
BDT$L_ACTION                      00000004     D
BDT$L_ADDR                        0000000C     D
BDT$L_CPUTYPE                     00000000     D
BDT$L_DEVTYPE                     00000002     D
BDT$L_DRIVRNAME                   00000014     D
BDT$L_ENTRY                       00000010     D
BDT$L_SIZE                        00000008     D
BOO$AL_VECTOR                     00000000 RG  D  02
BOO$DRIVER                      = 000001A2 RG  D  02
BOO$DRIVER_TBL                  = 00000000 R   D  03
BOO$MAP                           000000F1 RG  D  02
BOO$PURDPR                        00000149 R   D  02
BOO$QIO                           00000016 RG  D  02
BOO$QIOSIZ                      = 000001A2     D
BOO$SELECT                        00000000 RG  D  05
BTD$K_HSCCI                     = 00000020     D
BUF                             = 00000004     D
COMPUTE_PFN                       00000078 R   D  02
EXE$GB_ADPTYPE                    00000015 RG  D  02
EXE$GB_CPUTYPE                    00000014 RG  D  02
FILLSPT                           0000013B R   D  02
FUNC                            = 00000010     D
GET_BYTEOFFSET                    0000087 R    D  02
INIT_MAPREGS                      00000073 R   D  02
LBN                             = 0000000C     D
LOAD_MBA_MAP                      0000009F R   D  02
LOAD_NEXT_MAP                     000000A7 R   D  02
MBA$L_MAP                       = 00000800     D
MODE                            = 00000014     D
NDT$_MB                         = 00000020     D
PR$_MAPEN                       = 00000038     D
PR$_PQBR                        = 00000008     D
PR$_SID_TYP780                  = 00000001     D
PTE$C_KW                        = 10000000     D
PTE$M_PFN                       = 001FFFFF     D
PTE$M_VALID                     = 80000000     D
PUSH_RETRY                        000000C4 R   D  02
RPB                             = 0000000C     D
RPB$B_CONFREG                   = 00000090     D
RPB$B_DEVTYP                    = 00000066     D
RPB$L_ADPPHY                    = 0000005C     D
RPB$L_ADPVIR                    = 00000060     D
RPB$L_BOOTR1                    = 00000020     D
RPB$L_CSRPHY                    = 00000054     D
RPB$L_CSRVIR                    = 00000058     D
RPB$L_IOVEC                     = 00000034     D
RPB$L_IOVECSZ                   = 00000038     D
RPB$L_SVASPT                    = 00000050     D
SIZE                            = 00000008     D
SS$_NORMAL                      = 00000001     D
SS$_PARITY                      = 000001F4     D
SVASPT                          = 00000004     D
UBA$L_DPR                       = 00000040     D
UBA$L_MAP                       = 00000800     D
UBA$M_MAP_VALID                 = 80000000     D
UBA$V_DPR_BNE                   = 0000001F     D
UBA$V_DPR_XMTER                 = 0000001E     D
UBA$V_MAP_BO                    = 00000019     D
UBA$V_MAP_DPD                   = 00000015     D
UBI$C_PURCNT                    = 0000000A     D
UBI$L_DPR                       = 00000000     D
UBI$L_MAP                       = 00000800     D
UBI$L_SR                        = 00000010     D
UBI$M_SR_MRPE                   = 00008000     D
UBI$M_SR_NXM                    = 00010000     D
UBI$M_SR_UCE                    = 80000000     D
UBI$M_SR_UWE                    = 00004000     D
UBI$V_DPR_ERROR                 = 0000001F     D
UBI$V_DPR_PUR                   = 00000000     D
VA$S_BYTE                       = 00000009     D
VA$S_VPN                        = 00000015     D
VA$V_BYTE                       = 00000000     D
VA$V_SYSTEM                     = 0000001F     D
VA$V_VPN                        = 00000009     D
VABASE                          = 00000008     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000018 (24.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_1	000001A2 (418.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_3	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_5	00000004 (4.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_6	00000069 (105.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BDT$K_LENGTH	=00000018		#-561 (8)
BDT$L_ACTION	00000004		#-557 (8)
BDT$L_ADDR	0000000C		#-570.4 (8)
BDT$L_CPUTYPE	00000000		#-546 (8)
BDT$L_DEVTYPE	00000002		#-552 (8)
BDT$L_DRIVRNAME	00000014		#-570.6 (8)
BDT$L_ENTRY	00000010		#-570.3 (8)
BDT$L_SIZE	00000C08		#-570 (8)
BOO$AL_VECTOR	00000000-R	189 (3)	190 (3) 191 (3) 192 (3) 535.37 (7) #-570.1 (8) #-570.3 (8) #-570.4 (8) #-570.6 (8)
BOO$DRIVER	=000001A2-R	535.39 (7)	
BOO$DRIVER_IBL	=00000000-R	535.46 (7)	537.1 (8)
BOO$MAP	000000F1-R	459 (5)	191 (3)
BOO$PURDPR	00000149-R	535.1 (6)	#-426.4 (4)
BOO$QIO	00000016-R	253 (4)	190 (3)
BOO$QIOSIZ	=000001A2	535.37 (7)	#-569 (8)
BOO$SELECT	00000000-R	535.81 (8)	192 (3)
BTD$K_HSCCI	=00000020		#-288.1 (4)
BUF	=00000004	246 (4)	#-274 (4) #-361 (4)
COMPUTE_PFN	00000078-R	337 (4)	#-389 (4)
EXE$GB_ADPTYPE	00000015-R	209.1 (3)	#-287.1 (4) #-302.1 (4) #-343.1 (4) #-396.1 (4) #-426.1 (4)
EXE$GB_CPUTYPE	00000014-R	208 (3)	#-535.4 (6) #-537.3 (8)
FILLSPT	0000013B-R	486 (5)	#-469 (5) #-473 (5) #-489 (5)
FUNC	=00000010	249 (4)	
GET_BYTEOFFSET	000000B7-R	408 (4)	#-396.1 (4)
INIT_MAPREGS	00000073-R	332 (4)	#-302 (4)
LBN	=0000000C	248 (4)	#-283 (4)
LOAD_MBA_MAP	0000009F-R	376 (4)	#-343.1 (4)
LOAD_NEXT_MAP	000000A7-R	387 (4)	#-366 (4)
MBA$L_MAP	=00000800		333 (4) 334 (4) 335 (4)
MODE	=00000014	250 (4)	#-339 (4)
NDT$_MB	=00000020		#-301 (4)
PR$_MAPEN	=00000038		#-264 (4) #-418 (4)
PR$_POBR	=00000008		#-286 (4)
PR$_SID_TYP780	=00000001		#-535.4 (6)
PTE$C_KW	=10000000		#-487 (5)
PTE$M_PFN	=001FFFFF		#-341 (4)
PTE$M_VALID	=80000000		#-487 (5)
PUSH_RETRY	000000C4-R	421 (4)	#-288.2 (4)
RPB	=0000000C	452 (5)	#-263 (4) #-460 (5)
RPB$B_CONFREG	=00000090		#-300 (4)
RPB$B_DEVTYP	=00000066		#-288 (4) #-537.2 (8)
RPB$L_ADPPHY	=0000005C		265 (4) #-266 (4) 465 (5)
RPB$L_ADPVIR	=00000060		265 (4) #-464 (5)
RPB$L_BOOTR1	=00000020		#-299 (4)
RPB$L_CSRPHY	=00000054		416 (4) #-419 (4) #-471 (5) #-474 (5)
RPB$L_CSRVIR	=00000058		416 (4) #-475 (5)
RPB$L_IOVEC	=00000034		#-425 (4) #-578 (8)
RPB$L_IOVECSZ	=00000038		#-570 (8)

```
RPB$L_SVASPT                    =00000050                       #-284     (4)     #-462     (5)
SIZE                            =00000008       247     (4)     #-275     (4)
SS$_NORMAL                      =00000001                       #-535.3   (6)
SS$_PARITY                      =000001F4                       #-535.11  (6)
SVASPT                          =00000004       445     (5)     #-461     (5)
UBA$L_DPR                       =00000040                         535.6   (6)
UBA$L_MAP                       =00000800                         333     (4)
UBA$M_MAP_VALID                 =80000000                       #-363     (4)     #-398     (4)
UBA$V_DPR_BNE                   =0000001F                       #-535.7   (6)
UBA$V_DPR_XMTER                 =0000001E                       #-535.10  (6)     #-535.9   (6)
UBA$V_MAP_BO                    =00000019                       #-361     (4)
UBA$V_MAP_DPD                   =00000015                       #-364     (4)
UBI$C_PURCNT                    =0000000A                       #-535.16  (6)
UBI$L_DPR                       =00000000                         535.14  (6)
UBI$L_MAP                       =00000800                         334     (4)
UBI$L_SR                        =00000010                       #-535.26  (6)
UBI$M_SR_MRPE                   =00008000                       #-535.29  (6)
UBI$M_SR_NXM                    =00010000                       #-535.30  (6)
UBI$M_SR_UCE                    =80000000                       #-535.31  (6)
UBI$M_SR_UWE                    =00004000                       #-535.28  (6)
UBI$V_DPR_ERROR                 =0000001F                       #-535.22  (6)
UBI$V_DPR_PUR                   =00000000                       #-535.15  (6)     #-535.19  (6)
VA$S_BYTE                       =00000009                       #-278     (4)     #-409     (4)
VA$S_VPN                        =00000015                       #-287     (4)     #-465     (5)     #-467     (5)
VA$V_BYTE                       =00000000                       #-278     (4)     #-409     (4)
VA$V_SYSTEM                     =0000001F                       #-285     (4)
VA$V_VPN                        =00000009                       #-287     (4)     #-465     (5)     #-467     (5)
VABASE                          =00000008       448     (5)     #-463     (5)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BTDDEF	1	124 (2)	124 (2)
$DEFINI	1	124 (2)	124 (2) 125 (2) 126 (2) 127 (2) 128 (2) 129 (2) 130 (2) 131 (2) 132 (2) 133 (2) 134 (2) 167 (2)
$IODEF	17	125 (2)	125 (2)
$MBADEF	5	126 (2)	126 (2)
$NDTDEF	2	127 (2)	127 (2)
$PRDEF	4	128 (2)	128 (2)
$PTEDEF	3	129 (2)	129 (2)
$RPBDEF	5	130 (2)	130 (2)
$SSDEF	21	131 (2)	131 (2)
$UBADEF	6	132 (2)	132 (2)
$UBIDEF	2	133 (2)	133 (2)
$VADEF	1	134 (2)	134 (2)
ASSUME	1	265 (4)	265 (4) 333 (4) 334 (4) 416 (4)
CASE	1	535.4 (6)	535.4 (6)
CPUDISP	1	159 (2)	535.4 (6)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.12	00:00:00.25
Command processing	130	00:00:00.69	00:00:01.79
Pass 1	772	00:00:15.93	00:00:22.80
Symbol table sort	0	00:00:02.07	00:00:02.12
Pass 2	168	00:00:04.18	00:00:09.29
Symbol table output	10	00:00:00.09	00:00:00.09
Psect synopsis output	7	00:00:00.04	00:00:00.03
Cross-reference output	31	00:00:00.35	00:00:00.44
Assembler run totals	1157	00:00:23.48	00:00:36.82

The working set limit was 1000 pages.
71636 bytes (140 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1327 non-local and 24 local symbols.
689 source lines were read in Pass 1, producing 0 object records in Pass 2.
66 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	9
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	16

1410 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) BOOTDRIVR/UPDA=(BOOTDRIVR.UPD,BOOTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(1) 90 RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMBER
(1) 154 BOO$CACHE_INIT - INIT FILEREAD CACHE
(1) 287 BOO$IMAGE_ATT - Get image attributes from image header
(1) 333 SYS$ASSIGN, Dummy assign device system service

```
                        0000     1          .TITLE  BOOTIO - BOOTSTRAP FILEREAD IO MODULE
                        0000    .1          .IDENT  '06-02'
-1                      0000     3 ;
                        0000     4 ;*****************************************************************************
                        0000     5 ;*                                                                           *
                        0000    .1 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983 BY                                 *
-1                      0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                        0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
                        0000     9 ;*                                                                           *
                        0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                        0000    11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                        0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
                        0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000    14 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
                        0000    15 ;*  TRANSFERRED.                                                             *
                        0000    16 ;*                                                                           *
                        0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
                        0000    18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                        0000    19 ;*  CORPORATION.                                                             *
                        0000    20 ;*                                                                           *
                        0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                        0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000    23 ;*                                                                           *
                        0000    24 ;*                                                                           *
                        0000    25 ;*****************************************************************************
                        0000    26
                        0000    27 ;++
                        0000    28 ; FACILITY:     SYSTEM BOOTSTRAPPING
                        0000    29 ;
                        0000    30 ; ABSTRACT:
                        0000    31 ;
                        0000    32 ;       THIS MODULE PERFORMS LOGICAL BLOCK I/O FOR FILEREAD
                        0000    33 ;
                        0000    34 ; ENVIRONMENT:  KERNEL MODE, UNMAPPED, IPL=31
                        0000    35 ;
                        0000    36 ; AUTHOR: RICHARD I. HUSTVEDT   , CREATION DATE: 14-APR-78
                        0000    37 ;
                        0000    38 ; MODIFIED BY:
                        0000    39 ;
                        0000    .1 ;       02      Bob Bergazzi    June 29,1983    Version 6.12
                        0000    .2 ;               Adapted to run for supervisor.
                        0000    .3 ;
                        0000    40 ;       V03-001 KTA0091         Kerbey T. Altmann       30-Mar-1982
                        0000    41 ;               Redo some entries in the file cache table.  Make the
                        0000    42 ;               page allocation part of BOO$CACHE_INIT a subroutine.
                        0000    43 ;               Divide it into two pieces.
                        0000    44 ;
                        0000    45 ;       V02-003 PHL0012         Peter H. Lipman         02-Aug-1981
                        0000    46 ;               Explicitly initialize the FILEREAD cache pointer.
                        0000    47 ;       Must not allow assembly time initialization since VMB can restart.
                        0000    48 ;               Fix PSECT so that VMB can overlay most of itself with
                        0000    49 ;       the secondary boot file.  Remove BOO$GL_RPBBASE.
                        0000    50 ;
                        0000    51 ;       V02-002 PHL0011         Peter H. Lipman         31-Jul-1981
                        0000    52 ;               The page at which the FILEREAD cache is located
                        0000    53 ;       should be relative to the RPB not absolute.
                        0000    54 ;
```

```
                        0000     55 ;       V02-001 PHL0007         Peter H. Lipman         14-Mar-1981
                        0000     56 ;               Extend parmeter list to FIL$RDWRTLBN to make it
                        0000     57 ;       possible to read more than one block at a time.  This
                        0000     58 ;       enhancement was made in conjunction with the cacheing
                        0000     59 ;       feature in FILEREAD.
                        0000     60 ;               Add new BOO$CACHE_INIT routine for VMB and SYSBOOT to
                        0000     61 ;       init the FILEREAD cache.
                        0000     62 ;               Move other common FILEREAD pieces into this module thus
                        0000     63 ;       avoiding them being in two places.
                        0000     64 ;               Add new BOO$IMAGE_ATT routine to fetch certain image
                        0000     65 ;       attributes from an image header.
                        0000     66 ;
                        0000     67 ;--
```

```
                        0000     69 ;
                        0000     70 ; INCLUDE FILES:
                        0000     71 ;
                        0000     72         $IHDDEF                         ; IMAGE HEADER DEFINITIONS
                        0000     73         $IHSDEF                         ; IMAGE HEADER SYMBOL TABLE DEFS
                        0000     74         $IHPDEF                         ; IMAGE HEADER PATCH CONTROL DEFS
                        0000     75         $RPBDEF                         ; DEFINE RESTART PARAMETER BLOCK
                        0000     76 ;
                        0000     77 ; MACROS:
                        0000     78 ;
                        0000     79 ; Define Memory Size to File Cache Parameter table entry
                        0000     80 ;
                        0000     81         .MACRO  MEM_FILE_CACHE MEM_PAGE_CNT,CACHE_PAGE_NUM,CACHE_PAGE_CNT,MAX_PAGE
                        0000     82         .LONG   MEM_PAGE_CNT-<MEM_PAGE_CNT/10>
                        0000     83         .WORD   CACHE_PAGE_NUM
                        0000     84         .WORD   <<CACHE_PAGE_CNT+3>&^C<3>>
                        0000     85         .LONG   MAX_PAGE
                        0000     86         .ENDM   MEM_FILE_CACHE
```

```
                        00000000     .1          .PSECT  code, exe, nowrt, shr, long     ;                                [02]
-1                          0000     89
                            0000     90          .SBTTL  RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMBER
                            0000     91  ;++
                            0000     92  ; FUNCTIONAL DESCRIPTION:
                            0000     93  ;
                            0000     94  ;       THIS ROUTINE READS/WRITES N BYTES FROM/TO THE SPECIFIED
                            0000     95  ; LOGICAL BLOCK NUMBER OF THE VOLUME ASSIGNED TO THE SPECIFIED CHANNEL
                            0000     96  ;
                            0000     97  ; CALLING SEQUENCE:
                            0000     98  ;
                            0000     99  ;       CALLG   ARGLIST,FIL$RDWRITLBN
                            0000    100  ;
                            0000    101  ; INPUT PARAMETERS:
                            0000    102  ;
                            0000    103  ;       CHAN(AP)        =               ;CHANNEL ASSIGNED TO THE VOLUME TO READ
                            0000    104  ;       LBN(AP)         =               ;LOGICAL BLOCK NUMBER TO READ
                            0000    105  ;       BUFADR(AP)      =               ;ADDRESS OF BUFFER TO READ INTO
                            0000    106  ;       IOFUNC(AP)      =               ;I/O FUNCTION CODE
                            0000    107  ;       BYTCNT(AP)      =               ;NUMBER OF BYTES TO TRANSFER
                            0000    108  ;
                            0000    109  ; IMPLICIT INPUTS:
                            0000    110  ;
                            0000    111  ;       NONE
                            0000    112  ;
                            0000    113  ; OUTPUT PARAMETERS:
                            0000    114  ;
                            0000    115  ;       R0 = SYSTEM STATUS CODE
                            0000    116  ;
                            0000    117  ; IMPLICIT OUTPUTS:
                            0000    118  ;
                            0000    119  ;       NONE
                            0000    120  ;
                            0000    121  ; COMPLETION CODES:
                            0000    122  ;
                            0000    123  ;       NONE
                            0000    124  ;
                            0000    125  ; SIDE EFFECTS:
                            0000    126  ;
                            0000    127  ;       NONE
                            0000    128  ;
                            0000    129  ; EQUATED SYMBOLS:
                            0000    130  ;
                            0000    131  ;    OFFSETS FROM AP
                            0000    132  ;
                00000004    0000    133          CHAN            =       4       ;CHANNEL TO WHICH VOLUME IS ASSIGNED
                00000008    0000    134          LBN             =       8       ;LOGICAL BLOCK NUMBER
                0000000C    0000    135          BUFADR          =       12      ;BUFFER ADDRESS TO READ INTO
                00000010    0000    136          IOFUNC          =       16      ;FUNCTION CODE FOR THE QIO
                00000014    0000    137          BYTCNT          =       20      ;NUMBER OF BYTES TO TRANSFER
                            0000    138  ;
                            0000    139  ;--
                            0000    140
                            0000    141  FIL$RDWRTLBN::
                    0000    0000    142          .WORD   0
                  04 AC     DD      0002    143          PUSHL   CHAN(AP)                ; ADDRESS OF RPB
                50  6E      D0      0005    144          MOVL    (SP),R0                 ; GET ADDRESS OF RPB
```

```
                50    34 A0    D0  0008    145         MOVL    RPB$L_IOVEC(R0),R0      ; GET POINTER TO I/O ROUTINE VECTOR
                         00    DD  000C    146         PUSHL   #0                      ; SET MODE TO PHYSICAL ADDRESS
                      10 AC    DD  000E    147         PUSHL   IOFUNC(AP)              ; SET FUNCTION
                      08 AC    DD  0011    148         PUSHL   LBN(AP)                 ; LOGICAL BLOCK NUMBER
                      14 AC    DD  0014    149         PUSHL   BYTCNT(AP)              ; SET NUMBER OF BYTES
                      0C BC    DF  0017    150         PUSHAL  @BUFADR(AP)             ; SET BUFFER ADDRESS
              00 B040    06    FB  001A    151         CALLS   #6,@(R0)[R0]            ; CALL BOOTSTRAP DRIVER
                               04  001F    152         RET
```

```
                     0020   154          .SBTTL  BOO$CACHE_INIT - INIT FILEREAD CACHE
                     0020   155 ;++
                     0020   156 ;
                     0020   157 ; Functional description:
                     0020   158 ;
                     0020   159 ;       This routine establishes a desired FILEREAD cache size and
                     0020   160 ;       base address according to the size of memory.  It finds
                     0020   161 ;       good contiguous pages at or near the desired place and
                     0020   162 ;       calls the FIL$CACHE_INIT routine to initialize the cache.
                     0020   163 ;       The routine is further divided into two pieces: one to do
                     0020   164 ;       cache allocation, and one to do the actual mount and open.
                     0020   165 ;       This is necessary for VMB needs to allocate the cache long
                     0020   166 ;       befor it is ready to accept IO to the device.
                     0020   167 ;
                     0020   168 ; Calling Sequence:
                     0020   169 ;
                     0020   170 ;       BSBW    BOO$CACHE_INIT
                     0020   171 ;
                     0020   172 ; Inputs:
                     0020   173 ;
                     0020   174 ;       R11                         - RPB base address
                     0020   175 ;       RPB$L_PFNCNT(R11)           - actual number of good pages in memory
                     0020   176 ;       RPB$Q_PFNMAP+4(R11)         - base address of PFN bitmap
                     0020   177 ;
                     0020   178 ; Implicit inputs:
                     0020   179 ;
                     0020   180 ;       none
                     0020   181 ;
                     0020   182 ; Outputs:
                     0020   183 ;
                     0020   184 ;       R0-R4 altered
                     0020   185 ;       FIL$GQ_CACHE set up with size and address of cache
                     0020   186 ;
                     0020   187 ; Implicit outputs:
                     0020   188 ;
                     0020   189 ;--
                     0020   190 ;
                     0020   191 ; Table of memory sizes to file cache parameters
                     0020   192 ;
                     0020   193 ; NOTE: If this table is modified, a corresponding table in VMB around
                     0020   194 ;       label MEM_TAB should be checked for consitency.
                     0020   195 ;
                     0020    .1 ;MEM_CACHE_TABLE:                                                    begin   [2]
                     0020    .2 ;       MEM_FILE_CACHE 16384,2048,64,4096       ; More than 8 megabyte
                     0020    .3 ;       MEM_FILE_CACHE  8192,1024,64,2048       ; More than 4 megabyte
                     0020    .4 ;       MEM_FILE_CACHE  4096, 640,64,1024       ; More than 2 megabyte
                     0020    .5 ;       MEM_FILE_CACHE  2048, 512,64, 768       ; More than 1 megabyte
                     0020    .6 ;       MEM_FILE_CACHE  1024, 256,32, 512       ; More than 512k bytes
                     0020    .7 ;       MEM_FILE_CACHE   512, 256,16, 256       ; More than 256k bytes
                     0020    .8 ;       MEM_FILE_CACHE   384, 256, 8, 192       ; More than 192k bytes
                     0020    .9 ;       MEM_FILE_CACHE   256, 128, 4, 128       ; More than 128k bytes
                     0020   .10 ;       MEM_FILE_CACHE     0,   0, 0,   0       ;
```

```
                     0020   207 ;
                     0020   208 ; BOO$CACHE_ALLOC - The piece that does the allocation.
                     0020   209 ;
                     0020   210 ; Outputs:
                     0020   211 ;       FIL$GQ_CACHE filled in with size/address in blocks
                     0020   212 ;
                     0020    .1 ;BOO$CACHE_ALLOC::
                     0020    .2 ;       PUSHL   R5                          ; Save a register
                     0020    .3 ;       CLRQ    W^FIL$GQ_CACHE              ; Assume no cache available
                     0020    .4 ;       MOVAL   B^MEM_CACHE_TABLE,R0        ; Adr of memory size to cache params tbl
                     0020    .5 ;10$:   MOVQ    (R0)+,R1                    ; Get the next table entry
                     0020    .6 ;       BEQL    20$                         ; Branch if memory too small for cache
                     0020    .7 ;       MOVL    (R0)+,R5                    ; Max page
                     0020    .8 ;       CMPL    RPB$L_PFNCNT(R11),R1        ; More memory than this entry?
                     0020    .9 ;       BLSS    10$                         ; Branch if not, get next one
                     0020   .10 ;       MOVZWL  R2,R0                       ; Starting relative bit (page) in PFNMAP
                     0020   .11 ;       ASHL    #-16,R2,R1                  ; Count of bits (pages) to look for
                     0020   .12 ;       ASHL    #-1,R1,R4                   ; Settle for half if can't find all
                     0020   .13 ;       BSBB    BOO$ALLOC_PAGES             ; Go get the pages
                     0020   .14 ;       BLSS    20$                         ; Failed
                     0020   .15 ;       MOVQ    R2,W^FIL$GQ_CACHE           ; Success, record the values
                     0020   .16 ;20$:   POPL    R5                          ; Restore a register
                     0020   .17 ;       RSB
                     0020   .18 ;
-18                  0020   231 ;
                     0020   232 ; BOO$CACHE_INIT - Full routine to both allocate and open the cache
                     0020   233 ;
                     0020    .1 ;BOO$CACHE_INIT::
                     0020    .2 ;       BSBB    BOO$CACHE_ALLOC             ; Allocate the cache
                     0020    .3 ;                                           ; Fall thru to finish
                     0020    .4 ;
-4                   0020   238 ;
                     0020   239 ; BOO$CACHE_OPEN - Actually mount the device and fill the cache
                     0020   240 ;
                     0020    .1 ;BOO$CACHE_OPEN::
                     0020    .2 ;       MOVL    W^FIL$GQ_CACHE,R2           ; Pick up size
                     0020    .3 ;       BEQL    10$                         ; Zero length implies none
                     0020    .4 ;       SUBL    #4,SP                       ; Location to store channel
                     0020    .5 ;       MOVL    SP,R0                       ; Address to store channel
                     0020    .6 ;       SUBL3   #2,R2,-(SP)                 ; Blocks in directory LBN cache
                     0020    .7 ;       PUSHL   S^#<<1024-FIL$C_SIZE>/FIL$C_DIR_SIZE> ; No. of dir cache entries
                     0020    .8 ;       ASHL    #9,W^FIL$GQ_CACHE+4,-(SP)   ; Byte address from page number
                     0020    .9 ;       ASHL    #9,R2,-(SP)                 ; Size of cache in bytes
                     0020   .10 ;       CLRL    -(SP)                       ; Null device name string descriptor
                     0020   .11 ;       PUSHL   R0                          ; Address to store channel
                     0020   .12 ;       CALLS   #6,W^FIL$CACHE_INIT         ; Init the FIL$OPENFILE cache
                     0020   .13 ;                                           ; descriptor returned in FIL$GQ_CACHE
                     0020   .14 ;       ADDL    #4,SP                       ; Clean off channel
                     0020   .15 ;10$:   RSB
-15                  0020   256
                     0020   257 ;
                     0020   258 ; BOO$ALLOC_PAGES - Find a run of contiguous, good pages
                     0020   259 ;
                     0020   260 ; Inputs:
                     0020   261 ;       R0 - Page to start at
                     0020   262 ;       R1 - Number of pages needed
                     0020   263 ;       R4 - Number willing to settle for
```

```
                     0020   264 ;       R5 - Maximum page
                     0020   265 ; Outputs:
                     0020   266 ;       CC - Status (BLSS to an error routine)
                     0020   267 ;       R2 - Number found
                     0020   268 ;       R3 - Starting page number
                     0020   269 ;
                     0020    .1 ;BOO$ALLOC_PAGES::
                     0020    .2 ;       ROTL    #<32-9>,R11,R2          ; PFN of the RPB
                     0020    .3 ;       ADDL    R2,R0                   ; Convert relative PFN to absolute
                     0020    .4 ;       MOVL    R0,R3                   ; Make a copy of starting bit
                     0020    .5 ;30$:   CMPL    R5,R0                   ; Less than max page
                     0020    .6 ;       BLSS    50$                     ; No, failure
                     0020    .7 ;       BBS     R0,@RPB$Q_PFNMAP+4(R11),40$ ; Branch if this is a good page
                     0020    .8 ;       SUBL3   R3,R0,R2                ; Count of bits (pages) found
                     0020    .9 ;       CMPL    R2,R4                   ; Found enough?
                     0020   .10 ;       BGEQ    50$                     ; Branch if yes
                     0020   .11 ;       ADDL3   #1,R0,R3                ; No, reset starting base
                     0020   .12 ;40$:   INCL    R0                      ; Next bit (page)
                     0020   .13 ;       SOBGTR  R1,30$                  ; Branch if more to check
                     0020   .14 ;       SUBL3   R3,R0,R2                ; Count of bits (pages) found
                     0020   .15 ;       CMPL    R2,R4                   ; Found enough?
                     0020   .16 ;50$:   RSB                             ; Return (Status in CC)         end     [2]
```

```
                                0020   287          .SBTTL  BOO$IMAGE_ATT - Get image attributes from image header
                                0020   288 ;++
                                0020   289 ; Functional Description:
                                0020   290 ;
                                0020   291 ;       BOO$IMAGE_ATT returns to the caller some attributes of the image
                                0020   292 ;
                                0020   293 ; Calling Sequence:
                                0020   294 ;
                                0020   295 ;       BSBW    BOO$IMAGE_ATT
                                0020   296 ;
                                0020   297 ; Inputs:
                                0020   298 ;
                                0020   299 ;       R2 = Size of file in blocks
                                0020   300 ;       R3 = Address of image header block (first one only)
                                0020   301 ;
                                0020   302 ; Outputs:
                                0020   303 ;
                                0020   304 ;       R1 = Number of image header blocks at the front of the image
                                0020   305 ;       R2 = Size of image in blocks excluding the blocks at the end
                                0020   306 ;            containing local symbols, global symbols, or patch text
                                0020   307 ;
                                0020   308 ;--
                                0020   309
                                0020   310 BOO$IMAGE_ATT::
          50   04 A3    3C     0020   311          MOVZWL  IHD$W_SYMDBGOFF(R3),R0  ; ANY SYMBOL TABLE INFORMATION?
                 0D    13     0024   312          BEQL    20$                     ; BRANCH IF NOT
          51   6043    9E     0026   313          MOVAB   IHS$L_DSTVBN(R0)[R3],R1 ; ADR OF 1ST VBN IN DEBUG SYMBOL TABLE
                 19    10     002A   314          BSBB    40$                     ; PROCESS IT
          51   04 A043 9E     002C   315          MOVAB   IHS$L_GSTVBN(R0)[R3],R1 ; ADR OF 1ST VBN IN GLOBAL SYMBOL TABLE
                 12    10     0031   316          BSBB    40$                     ; PROCESS IT
          50   08 A3    3C     0033   317 20$:     MOVZWL  IHD$W_PATCHOFF(R3),R0   ; ANY PATCH CONTROL INFORMATION?
                 07    13     0037   318          BEQL    30$                     ; BRANCH IF NOT
          51   20 A043 9E     0039   319          MOVAB   IHP$L_PATCOMTXT(R0)[R3],R1 ; ADR OF 1ST VBN OF PATCH COMMAND TEXT
                 05    10     003E   320          BSBB    40$                     ; PROCESS IT
          51   10 A3    9A     0040   321 30$:     MOVZBL  IHD$B_HDRBLKCNT(R3),R1  ; GET IMAGE HEADER BLOCK COUNT
                       05     0044   322          RSB
                              0045   323 ;
                              0045   324 ; SEE IF VBN IS NON ZERO AND THEN IF IT IS SMALLER THAN THE CURRENT SMALLEST
                              0045   325 ;
          51   61    01    C3     0045   326 40$:     SUBL3   #1,(R1),R1              ; FETCH VBN - 1
                 08    19     0049   327          BLSS    50$                     ; BRANCH IF NO VBN IS PRESENT
               51    52    D1     004B   328          CMPL    R2,R1                   ; IS IT SMALLER THAN THE CURRENT ONE
                 03    15     004E   329          BLEQ    50$                     ; BRANCH IF NOT
               52    51    D0     0050   330          MOVL    R1,R2                   ; YES, USE IT
                       05     0053   331 50$:     RSB
```

```
                        0054   333          .SBTTL  SYS$ASSIGN, Dummy assign device system service
                        0054   334
                        0054   335 ;++
                        0054   336 ;
                        0054   337 ; Functional description:
                        0054   338 ;
                        0054   339 ;       SYS$ASSIGN is a dummy routine to satisfy the requirements of
                        0054   340 ;       FIL$OPENFILE.
                        0054   341 ;
                        0054   342 ; Inputs:
                        0054   343 ;
                        0054   344 ;       CHAN(AP)        - address at which to return channel
                        0054   345 ;
                        0054   346 ; Implicit inputs:
                        0054   347 ;
                        0054   348 ;       The label BOO$GL_RPBBASE contains the physical address of the RPB.
                        0054   349 ;
                        0054   350 ; Outputs:
                        0054   351 ;
                        0054   352 ;       R0      - success status code
                        0054   353 ;
                        0054   354 ; Implicit outputs:
                        0054   355 ;
                        0054   356 ;       The channel returned is not a channel. It is instead the base
                        0054   357 ;       address of the RPB.
                        0054   358 ;
                        0054   359 ;--
                        0054    .1 ;                                                            begin   [2]
                        0054    .2 ;CHAN   = 8
                        0054    .3 ;
                        0054    .4 ;SYS$ASSIGN::                                ; Dummy system service.
                        0054    .5 ;        .WORD   0
                        0054    .6 ;
                        0054    .7 ;        MOVL    W^BOO$GL_RPBBASE,@CHAN(AP) ; Store RPB address as channel.
                        0054    .8 ;        MOVL    S^#SS$_NORMAL,R0            ; Return success status.
                        0054    .9 ;        RET                                 ; Return to caller.
                        0054   .10 ;        .PAGE
                        0054   .11 ;        .SBTTL  Common Globals for VMB and SYSBOOT
-11                     0054   371 ;
                        0054   372 ; The following globals are common to VMB and SYSBOOT and are
                        0054   373 ; defined here to avoid replicate definitions.
                        0054   374 ;
                        0054    .1 ;FIL$GT_DDSTRING::                           ; Default directory string.
                        0054    .2 ;        .ASCIC  /[SYSEXE]/
                        0054    .3 ;FIL$GT_DDDEV::                              ; Default device name
                        0054    .4 ;        .BYTE   0                           ; Null ASCIC string
-4                      0054   379
                        0054   380          .END
```

```
BOO$IMAGE_ATT                     00000020 RG D  02
BUFADR                          = 0000000C    D
BYTCNT                          = 00000014    D
CHAN                            = 00000004    D
FIL$RDWRTLBN                      00000000 RG D  02
IHD$B_HDRBLKCNT                 = 00000010    D
IHD$W_PATCHOFF                  = 00000008    D
IHD$W_SYMDBGOFF                 = 00000004    D
IHP$L_PATCOMTX1                 = 00000020    D
IHS$L_DSTVBN                    = 00000000    D
IHS$L_GSTVBN                    = 00000004    D
IOFUNC                          = 00000010    D
LBN                             = 00000008    D
RPB$L_IOVEC                     = 00000034    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	00000054 (84.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

+-------------------------+
! Symbol Cross Reference !
+-------------------------+

SYMBOL	VALUE	DEFINITION	REFERENCES...
BOO$IMAGE_ATT	00000020-R	310 (1)	
BUFADR	=0000000C	135 (1)	150 (1)
BYTCNT	=00000014	137 (1)	#-149 (1)
CHAN	=00000004	133 (1)	#-143 (1)
FIL$RDWRTLBN	00000000-R	141 (1)	
IHD$B_HDRBLKCNT	=00000010		#-321 (1)
IHD$W_PATCHOFF	=00000008		#-317 (1)
IHD$W_SYMDBGOFF	=00000C04		#-311 (1)
IHP$L_PATCOMTXT	=00000020		319 (1)
IHS$L_DSTVBN	=00000000		313 (1)
IHS$L_GSTVBN	=00000004		315 (1)
IOFUNC	=00000010	136 (1)	#-147 (1)
LBN	=00000008	134 (1)	#-148 (1)
RPB$L_IOVEC	=00000034		#-145 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...					
$DEFINI	1	72 (1)	72 (1)	73 (1)	74 (1)	75 (1)		
$!HDDEF	3	72 (1)	72 (1)					
$IHPDEF	1	74 (1)	74 (1)					
$IHSDEF	1	73 (1)	73 (1)					
$RPBDEF	5	75 (1)	75 (1)					
MEM_FILE_CACHE	1	81 (1)						

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.15	00:00:00.27
Command processing	133	00:00:00.77	00:00:02.30
Pass 1	271	00:00:03.96	00:00:06.46
Symbol table sort	0	00:00:00.20	00:00:00.21
Pass 2	91	00:00:01.53	00:00:02.90
Symbol table output	3	00:00:00.02	00:00:00.02
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	13	00:00:00.09	00:00:00.09
Assembler run totals	553	00:00:06.76	00:00:12.28

The working set limit was 1000 pages.
13295 bytes (26 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 183 non-local and 4 local symbols.
383 source lines were read in Pass 1, producing 0 object records in Pass 2.
25 pages of virtual memory were used to define 11 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	7

247 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) BOOTIO/UPDA=(BOOTIO.UPD,BOOTIO.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 51 GET ONE BYTE OF DATA FROM USER BUFFER
(1) 87 PUT ONE BYTE OF DATA INTO USER'S BUFFER
(1) 121 INITIALIZE FOR SINGLE BYTE TRANSFERS
(1) 143 MOVE FROM USER BUFFER
(1) 176 MOVE TO USER BUFFER
(1) 209 FILL SYSTEM PTE WITH TRANSFER PTE

```
V54                     0000     .1          .TITLE  BUFCTL  *** BUFCTL QIO buffer control
V54                     0000     .2          .IDENT  /05-03/
-3                      0000      4
                        0000      5 ;
                        0000      6 ;*****************************************************************************
                        0000      7 ;*                                                                           *
                        0000      8 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      9 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
                        0000     10 ;*                                                                           *
                        0000     11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000     12 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
                        0000     13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
                        0000     14 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000     15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
                        C000     16 ;*  TRANSFERRED.                                                             *
                        0000     17 ;*                                                                           *
                        0000     18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
                        0000     19 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                        0000     20 ;*  CORPORATION.                                                             *
                        0000     21 ;*                                                                           *
                        0000     22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000     23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000     24 ;*                                                                           *
                        0000     25 ;*****************************************************************************
                        0000     26 ;
                        0000     27 ; D. N. CUTLER 9-AUG-76
                        0000     28 ;
                        0000     29 ; MODIFIED BY:
                        0000     30 ;
V54                     0000     .1 ;                NICK HOWGATE    28-MAR-78       VERSION 02 (ESSAA-4.01).
V54                     0000     .2 ;       01       INCLUDED IN DIAGNOSTIC SUPERVISOR FOR QIO SUPPORT
V54                     0000     .3 ;                Roger Riggs     9 April 1979    Version 5.0
V54                     0000     .4 ;       02       Changed MOVFRUSER and MOVTOUSER not to use SVPN,
V54                     0000     .5 ;                instead just address it directly.
V54                     0000     .6 ;                Dave Butenhof   13-may-1980, Version 5.4
V54                     0000     .7 ;       03       Upgrade to VMS V2.0 level: add routines IOC$GETBYTE and
V54                     0000     .8 ;                IOC$PUTBYTE, dummy IOC$INITBUFWIND and IOC$FILSPT for
V54                     0000     .9 ;                code centralization, and extra entry points to
V54                     0000    .10 ;                IOC$MOVTOUSER and IOC$MOVFRUSER.
                        0000     31 ;       04       STJ0002         S. JEFFREYS             29-FEB-1980
                        0000     32 ;       ADD ALTERNATE ENTRY POINTS FOR IOC$MOVTOUSER AND IOC$MOVFRUSER.
                        0000     33 ;
                        0000     34 ;       03       STJ0001         S. JEFFREYS             26-SEP-1979
                        0000     35 ;       MODIFY IOC$PUTBYTE AND IOC$GETBYTE TO WORK CORRECTLY FOR
                        0000     36 ;       BUFFERS THAT ARE PAGE ALIGNED.
                        0000     37 ;
                        0000     38 ;       02       CAM001          C. MONIA                15-FEB-1979
                        0000     39 ;       ADD IOC$PUTBYTE AND IOC$GETBYTE ROUTINES FOR TU-58 SUPPORT
                        0000     40 ;
                        0000     41 ;
                        0000     42 ; I/O BUFFER CONTROL ROUTINES
                        0000     43 ;
                        0000     44 ; MACRO LIBRARY CALLS
                        0000     45 ;
                        0000     46
                        0000     47          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     48          $PTEDEF                         ;PAGE TABLE ENTRY DEFINITIONS
```

```
                                0000     49          $UCBDEF                         ;DEFINE UCB OFFSETS
V54                             0000     .1 ;
V54                             0000     .2 ; OWN STORAGE
V54                             0000     .3 ;
V54                         00000000     .4          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
```

```
                                0000     51          .SBTTL  GET ONE BYTE OF DATA FROM USER BUFFER
                                0000     52 ;+
                                0000     53 ; IOC$GETBYTE - GET ONE BYTE OF DATA FROM USER'S BUFFER
                                0000     54 ;
                                0000     55 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO GET A SINGLE BYTE  FROM THE
                                0000     56 ; USER'S BUFFER.
                                0000     57 ;
                                0000     58 ; PRIOR TO CALLING THIS ROUTINE, A CALL TO IOC$INITBUFWIND MUST BE MADE TO
                                0000     59 ; MAP THE SYSTEM PAGE TABLE ENTRY INTO THE USER'S BUFFER
                                0000     60 ;
                                0000     61 ; INPUTS:
                                0000     62 ;
                                0000     63 ;       R0 = SYSTEM VIRTUAL ADDRESS OF ONE-PAGE WINDOW INTO USER'S BUFFER.
                                0000     64 ;       R5 = ADDRESS OF UCB.
                                0000     65 ;
                                0000     66 ; OUTPUTS:
                                0000     67 ;
                                0000     68 ;       R0 = UPDATED SYSTEM VIRTUAL ADDRESS
                                0000     69 ;       R1 = ONE BYTE OF DATA (ZERO EXTENDED)
                                0000     70 ;
                                0000     71 ;       UCB$L_SVAPTE IS UPDATED WHENEVER A PAGE BOUNDARY IS CROSSED
                                0000     72 ;
                                0000     73 ; THE DRIVER IS EXPECTED TO SAVE THE VALUE OF R0 FOR SUBSEQUENT CALLS.
                                0000     74 ;
                                0000     75 ;-
                                0000     76
V54                             0000     .1
-1                              0000     78
                                0000     79 IOC$GETBYTE::                                ;
      51   80   90               0000     80          MOVB    (R0)+,R1                   ;GET BYTE FROM USER'S BUFFER
50    01FF 8F   B3               0003     81          BITW    #^X01FF,R0                 ;OVERFLOW PAGE BOUNDARY?
           06   12               0008     82          BNEQ    10$                        ;IF NEQ NO
  68 A5    04   C0               000A     83          ADDL    #4,UCB$L_SVAPTE(R5)        ;UPDATE ADDRESS OF PROCESS PTE
           49   10               000E     84          BSBB    IOC$FILSPT                 ;FILL SPT, COMPUTE SYSTEM ADDRESS OF PAGE
                05               0010     85 10$:     RSB                                ;RETURN
```

```
                                0011    87          .SBTTL  PUT ONE BYTE OF DATA INTO USER'S BUFFER
                                0011    88 ;+
                                0011    89 ; IOC$PUTBYTE - PUT ONE BYTE OF DATA IN USER'S BUFFER
                                0011    90 ;
                                0011    91 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO PUT A SINGLE BYTE OF DATA
                                0011    92 ; INTO THE USER'S BUFFER.
                                0011    93 ;
                                0011    94 ; PRIOR TO CALLING THIS ROUTINE, A CALL TO IOC$INITBUFWIND MUST BE MADE TO
                                0011    95 ; MAP THE SYSTEM PAGE TABLE ENTRY INTO THE USER'S BUFFER.
                                0011    96 ;
                                0011    97 ; INPUTS:
                                0011    98 ;
                                0011    99 ;       R0 = SYSTEM VIRTUAL ADDRESS OF ONE-PAGE WINDOW INTO USER'S BUFFER
                                0011   100 ;       R1 LOW BYTE = DATA TO BE TRANSFERRED TO USER
                                C011   101 ;       R5 = ADDRESS OF UCB
                                0011   102 ;
                                0011   103 ; OUTPUTS:
                                0011   104 ;
                                0011   105 ;       R0 = UPDATED SYSTEM VIRTUAL ADDRESS OF BUFFER WINDOW
                                0011   106 ;
                                0011   107 ;       UCB$L_SVAPTE IS UPDATED WHENEVER A PAGE BOUNDARY IS CROSSED
                                0011   108 ;
                                0011   109 ; THE DRIVER IS EXPECTED TO SAVE THE VALUE OF R0 FOR SUBSEQUENT CALLS.
                                0011   110 ;
                                0011   111 ;-
                                0011   112
                                0011   113 IOC$PUTBYTE::                                ;
     80   51    90              0011   114          MOVB    R1,(R0)+                    ;PUT BYTE INTO USERS'S BUFFER
50   01FF 8F    B3              0014   115          BITW    #^X01FF,R0                  ;OVERFLOW PAGE BOUNDARY?
          06    12              0019   116          BNEQ    10$                         ;IF NEQ NO
  68 A5   04    C0              001B   117          ADDL    #4,UCB$L_SVAPTE(R5)         ;UPDATE ADDRESS OF PROCESS PTE
          38    10              001F   118          BSBB    IOC$FILSPT                  ;FILL SPT, COMPUTE SYSTEM ADDRESS OF PAGE
          05                    0021   119 10$:     RSB                                 ;RETURN
```

```
                                0022   121          .SBTTL  INITIALIZE FOR SINGLE BYTE TRANSFERS
                                0022   122 ;+
                                0022   123 ; IOC$INITBUFWIND - INITIALIZE ONE-PAGE WINDOW INTO USER'S BUFFER
                                0022   124 ;
                                0022   125 ; THIS ROUTINE MUST BE CALLED BY A DRIVER TO SETUP THE INITIAL ONE-
                                0022   126 ; PAGE WINDOW INTO A USER'S BUFFER BEFORE CALLING IOC$GETBYTE OR
                                0022   127 ; IOC$PUTBYTE.
                                0022   128 ;
                                0022   129 ; INPUTS:
                                0022   130 ;
                                0022   131 ;       R5 = ADDRESS OF UCB
                                0022   132 ;
                                0022   133 ; OUTPUTS:
                                0022   134 ;
                                C022   135 ;       R0 = SYSTEM VIRTUAL ADDRESS OF WINDOW INTO USER'S BUFFER
                                0022   136 ;-
                                0022   137
                                0022   138 IOC$INITBUFWIND::                              ;
                          35    10    0022   139          BSBB    IOC$FILSPT              ;FILL SPT, COMPUTE VIRTUAL ADDRESS OF PAGE
                    50    6C A5     A8  0024   140          BISW    UCB$W_BOFF(R5),R0       ;MERGE BYTE OFFSET INTO ADDRESS
                                05    0028   141          RSB                             ;
```

```
                                0029   143          .SBTTL  MOVE FROM USER BUFFER
                                0029   144 ;+
                                0029   145 ; IOC$MOVFRUSER - MOVE FROM USER BUFFER
                                0029   146 ;
                                0029   147 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO MOVE A STRING FROM A USER
                                0029   148 ; BUFFER TO AN INTERNAL BUFFER.
                                0029   149 ;
                                0029   150 ; INPUTS:
                                0029   151 ;
                                0029   152 ;       R1 = ADDRESS OF INTERNAL BUFFER.
                                0029   153 ;       R2 = NUMBER OF BYTES TO BE MOVED.
                                0029   154 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0029   155 ;
                                0029   156 ; OUTPUTS:
                                C029   157 ;
                                0029   158 ;       ***TBS***
                                0029   159 ;-
                                0029   160
                                0029   161          .ENABLE LSB
                                0029   162 IOC$MOVFRUSER::                                ;MOVE FROM USER BUFFER
                     F7   10    0029   163          BSBB    IOC$INITBUFWIND              ;SETUP WINDOW INTO BUFFER
                     0D   11    002B   164          BRB     20$                          ;
                                002D   165 IOC$MOVFRUSER2::
           50  01FF   8F   B3   002D   166 10$:     BITW    #^X01FF,R0                   ;OVERFLOW PAGE BOUNDRY?
                     06   12    0032   167          BNEQ    20$                          ;IF NEQ NO
             68 A5   04   C0    0034   168          ADDL    #4,UCB$L_SVAPTE(R5)          ;UPDATE ADDRESS OF USER PTE
                     1F   10    0038   169          BSBB    IOC$FILSPT                   ;FILL SYSTEM PTE WITH PROPER RELOCATION
                                003A   170 IOC$MOVFRUSER1::
           81   80   90    003A   171 20$:     MOVB    (R0)+,(R1)+                  ;MOVE BYTE TO INTERNAL BUFFER
             ED 52   F5    003D   172          SOBGTR  R2,10$                       ;ANY MORE BYTES TO MOVE?
                     05    0040   173          RSB                                  ;
                                0041   174          .DSABL  LSB
```

```
                                0041   176          .SBTTL  MOVE TO USER BUFFER
                                0041   177 ;+
                                0041   178 ; IOC$MOVTOUSER - MOVE TO USER BUFFER
                                0041   179 ;
                                0041   180 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO MOVE A STRING FROM AN INTERNAL
                                0041   181 ; BUFFER TO A USER BUFFER.
                                0041   182 ;
                                0041   183 ; INPUTS:
                                0041   184 ;
                                0041   185 ;       R1 = ADDRESS OF INTERNAL BUFFER.
                                0041   186 ;       R2 = NUMBER OF BYTES TO BE MOVED.
                                0041   187 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0041   188 ;
                                0041   189 ; OUTPUTS:
                                0041   190 ;
                                0041   191 ; ***TBS***
                                0041   192 ;-
                                0041   193
                                0041   194          .ENABLE LSB
                                0041   195 IOC$MOVTOUSER::                          ;MOVE TO USER BUFFER
                    DF   10     0041   196          BSBB    IOC$INITBUFWIND        ;INITIALIZE WINDOW INTO BUFFER
                    0D   11     0043   197          BRB     20$                    ;
                                0045   198 IOC$MOVTOUSER2::
              50    01FF 8F     B3     0045   199 10$:    BITW    #^X01FF,R0             ;OVERFLOW PAGE BOUNDRY?
                    06   12     004A   200          BNEQ    20$                    ;IF NEQ NO
                68  A5   04     C0     004C   201          ADDL    #4,UCB$L_SVAPTE(R5)    ;UPDATE ADDRESS OF USER PTE
                    07   10     0050   202          BSBB    IOC$FILSPT             ;FILL SYSTEM PTE WITH PROPER RELOCATION
                                0052   203 IOC$MOVTOUSER1::
                    80   81     90     0052   204 20$:    MOVB    (R1)+,(R0)+            ;MOVE BYTE TO USER BUFFER
                      ED 52     F5     0055   205          SOBGTR  R2,10$              ;ANY MORE BYTES TO MOVE?
                    05   10     0058   206          RSB                            ;
                                0059   207          .DSABL  LSB
```

```
                                0059   209          .SBTTL  FILL SYSTEM PTE WITH TRANSFER PTE
                                0059   210 ;+
                                0059   211 ; IOC$FILSPT - FILL SYSTEM PTE WITH TRANSFER PTE
                                0059   212 ;
V54                             0059    .1 ; This routine extracts the PFN from current PTE and creates a virtual
V54                             0059    .2 ; address from it.
-2                              0059   215 ;
                                0059   216 ; INPUTS:
                                0059   217 ;
                                0059   218 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0059   219 ;
                                0059   220 ; OUTPUTS:
                                0059   221 ;
                                0059   222 ;       R0 = SYSTEM VIRTUAL ADDRESS OF START OF PAGE CONTAINING THE BUFFER.
                                0059   223 ;
                                0059   224 ;       REGISTERS R1, R2, AND R3 ARE PRESERVED ACROSS CALL.
                                0059   225 ;-
                                0059   226
                                0059   227 IOC$FILSPT::                                ;FILL SYSTEM PTE WITH TRANSFER PTE
V54                 15    00    EF    0059    .1          EXTZV   #PTE$V_PFN, #PTE$S_PFN, -
                  50    68 B5         005C
V54                                     005F    .2                   @UCB$L_SVAPTE(R5), R0  ; Get PFN of user buffer
V54               50    50    09    78  005F    .3          ASHL    #9, R0, R0             ; Re-make virtual address
-11                           05    0063   239          RSB                                ;
                                0064   240
                                0064   241          .END
```

```
IOC$FILSPT          00000059 RG D  02      UCB$L_DPC           00000080  D      UCB$W_FUNC          0000007E  D
IOC$GETBYTE         00000000 RG D  02      UCB$L_DUETIM        0000005C  D      UCB$W_MB_SEED       00000000  D
IOC$INITBUFWIND     00000022 RG D  02      UCB$L_DX_BFPNT      0000009C  D      UCB$W_MSGCNT        00000016  D
IOC$MOVFRUSER       00000029 RG D  02      UCB$L_DX_BUF        00000098  D      UCB$W_MSGMAX        00000014  D
IOC$MOVFRUSER1      0000003A RG D  02      UCB$L_DX_RXDB       000000A0  D      UCB$W_NT_CHAN       0000007C  D
IOC$MOVFRUSER2      0000002D RG D  02      UCB$L_EMB           00000078  D      UCB$W_OFFSET        0000008A  D
IOC$MOVTOUSER       00000041 RG D  02      UCB$L_FIRST         00000014  D      UCB$W_REFC          00000050  D
IOC$MOVTOUSER1      00000052 RG D  02      UCB$L_FPC           0000000C  D      UCB$W_SIZE          00000008  D
IOC$MOVTOUSER2      00000045 RG D  02      UCB$L_FQBL          00000004  D      UCB$W_SRCADDR       0000001A  D
IOC$PUTBYTE         00000011 RG D  02      UCB$L_FQFL          00000000  D      UCB$W_STS           00000058  D
PTE$S_PFN       =   00000015       D      UCB$L_FR3           00000010  D      UCB$W_TT_DESIZE     000000A5  D
PTE$V_PFN       =   00000000       D      UCB$L_FR4           00000014  D      UCB$W_UNIT          00000048  D
SIZ...          =   00000002       D      UCB$L_IOQBL         00000044  D      UCB$W_VPROT         0000001A  D
UCB$B_AMOD          00000053       D      UCB$L_IOQFL         00000040  D
UCB$B_CEX           00000077       D      UCB$L_IRP           0000004C  D
UCB$B_CM1           0000004A       D      UCB$L_LINK          0000002C  D
UCB$B_CM2           0000004B       D      UCB$L_LOGADR        00000064  D
UCB$B_DEVCLASS      00000038       D      UCB$L_MAXBLOCK      00000084  D
UCB$B_DEVTYPE       00000039       D      UCB$L_MB_MBX        0000007C  D
UCB$B_DIPL          00000052       D      UCB$L_MB_PORT       0000008C  D
UCB$B_DX_SCTCNT     000000A6       D      UCB$L_MB_RAST       00000078  D
UCB$B_ERTCNT        00000070       D      UCB$L_MB_SHB        00000080  D
UCB$B_ERTMAX        00000071       D      UCB$L_MB_WAST       00000074  D
UCB$B_FEX           00000076       D      UCB$L_MB_WIOQBL     00000088  D
UCB$B_FIPL          0000000B       D      UCB$L_MB_WIOQFL     00000084  D
UCB$B_LOCSRV        0000003C       D      UCB$L_MEDIA         0000008C  D
UCB$B_OFFNDX        00000094       D      UCB$L_NT_DATSSB     00000074  D
UCB$B_OFFRTC        00000095       D      UCB$L_NT_INTSSB     00000078  D
UCB$B_REMSRV        0000003D       D      UCB$L_OPCNT         00000060  D
UCB$B_SECTORS       0000003C       D      UCB$L_OWNUIC        0000001C  D
UCB$B_SLAVE         00000074       D      UCB$L_PID           00000028  D
UCB$B_SPR           00000075       D      UCB$L_RQBL          00000004  D
UCB$B_STATE         00000052       D      UCB$L_RQFL          00000000  D
UCB$B_TRACKS        0000003D       D      UCB$L_SVAPTE        00000068  D
UCB$B_TT_CRFILL     0000009D       D      UCB$L_SVPN          00000064  D
UCB$B_TT_DECRF      000000A1       D      UCB$L_TT_DECHAR     000000A8  D
UCB$B_TT_DELFF      000000A2       D      UCB$L_TT_RDUE       0000008C  D
UCB$B_TT_DESPEE     000000A0       D      UCB$L_TT_RTIMOU     000000B8  D
UCB$B_TT_DETYPE     000000A4       D      UCB$L_VCB           00000030  D
UCB$B_TT_LFFILL     0000009E       D      UCB$T_PARTNER       0000000C  D
UCB$B_TT_SPEED      0000009C       D      UCB$W_BCNT          0000006E  D
UCB$B_TYPE          0000000A       D      UCB$W_BCR           00000096  D
UCB$B_VERTSZ        0000003F       D      UCB$W_BOFF          0000006C  D
UCB$C_LENGTH        00000074       D      UCB$W_BUFQUO        00000018  D
UCB$C_MB_LENGTH     00000090       D      UCB$W_BYTESTOGO     0000003E  D
UCB$C_TT_LENGTH     000000BC       D      UCB$W_CHARGE        0000004A  D
UCB$K_LENGTH        00000074       D      UCB$W_CYLINDERS     0000003E  D
UCB$K_MB_LENGTH     00000090       D      UCB$W_DA            0000008C  D
UCB$K_TT_LENGTH     000000BC       D      UCB$W_DC            0000008E  D
UCB$L_AMB           00000054       D      UCB$W_DEVBUFSIZ     0000003A  D
UCB$L_ASTQBL        00000010       D      UCB$W_DEVSTS        0000005A  D
UCB$L_ASTQFL        0000000C       D      UCB$W_DIRSEQ        00000088  D
UCB$L_CPID          0000005C       D      UCB$W_DSTADDR       00000018  D
UCB$L_CRB           00000020       D      UCB$W_DX_BCR        000000A4  D
UCB$L_DDB           00000024       D      UCB$W_EC1           00000090  D
UCB$L_DEVCHAR       00000034       D      UCB$W_EC2           00000092  D
UCB$L_DEVDEPEND     0000003C       D      UCB$W_ERRCNT        00000072  D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000064 (100.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
IOC$FILSPT	00000059-R	227 (1)	#-118 (1) #-139 (1) #-169 (1) #-202 (1) #-84 (1)
IOC$GETBYTE	00000000-R	79 (1)	
IOC$INITBUFWIND	00000022-R	138 (1)	#-163 (1) #-196 (1)
IOC$MOVFRUSER	00000029-R	162 (1)	
IOC$MOVFRUSER1	0000003A-R	170 (1)	
IOC$MOVFRUSER2	0000002D-R	165 (1)	
IOC$MOVTOUSER	00000041-R	195 (1)	
IOC$MOVTOUSER1	00000052-R	203 (1)	
IOC$MOVTOUSER2	00000045-R	198 (1)	
IOC$PUTBYTE	00000011-R	113 (1)	
PTE$S_PFN	=00000015		#-227.1 (1)
PTE$V_PFN	=00000000		#-227.1 (1)
UCB$L_SVAPTE	00000068		#-117 (1) #-168 (1) #-201 (1) 227.2 (1) #-83 (1)
UCB$W_BOFF	0000006C		#-140 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...			
$DEFINI	1	47 (1)	47 (1)	48 (1)	49 (1)	
$PRDEF	4	47 (1)	47 (1)			
$PTEDEF	3	48 (1)	48 (1)			
$UCBDEF	10	49 (1)	49 (1)			

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.23
Command processing	140	00:00:00.72	00:00:01.48
Pass 1	284	00:00:05.88	00:00:08.55
Symbol table sort	0	00:00:00.45	00:00:00.66
Pass 2	81	00:00:01.27	00:00:02.22
Symbol table output	11	00:00:00.11	00:00:00.27
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	7	00:00:00.09	00:00:00.09
Assembler run totals	562	00:00:08.67	00:00:13.52

The working set limit was 1000 pages.
23486 bytes (46 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 327 non-local and 6 local symbols.
246 source lines were read in Pass 1, producing 0 object records in Pass 2.
25 pages of virtual memory were used to define 12 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	8

504 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) BUFCTL/UPDA=(BUFCTL.UPD,BUFCTL.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

'able of contents

(1) 55 DECLARATIONS
(2) 82 GET A VERTUAL MEMORY BUFFER.
(4) 161 RELEASE A MEMORY BUFFER.

```
0000      1          .TITLE  BUFFER  *** BUFFER GETBUF/RELBUF routines
0000      2          .IDENT  /07-06/
0000      3          .LIST   MEB
0000      4          .NLIST  CND
0000      5
0000      6 ;
0000      7 ; COPYRIGHT (C) 1977, 1980, 1983
0000      8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      9 ;
0000     10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
C000     15 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     16 ; REMAIN IN DEC.
0000     17 ;
0000     18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     20 ; CORPORATION.
0000     21 ;
0000     22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     24
0000     25 ;++
0000     26 ; FACILITY:      VAX DIAGNOSTIC SUPERVISOR.
0000     27 ;
0000     28 ; ABSTRACT:
0000     29 ;
0000     30 ; ENVIRONMENT:
0000     31 ;
0000     32 ; AUTHOR:        KEN CHAPMAN      10-NOV-77       VERSION 01.
0000     33 ;
0000     34 ; MODIFIED BY:
0000     35 ;                KEN CHAPMAN      24-MAR-78       VERSION 02 (ESSAA-4.01).
0000     36 ;       01       ADDED 'REGION' ARGUMENT TO ''GETBUF'' AND ''RELBUF''.
0000     37 ;       02       PUT PAGE COUNT CHECK IN ''RELBUF''.
0000     38 ;
0000     39 ;       03       - Jack Stansbury, 21-Oct-1981, Version 6.-
0000     40 ;                Added .LIBRARY statements for $DIAG and $DS.
0000     41 ;
0000     42 ;       04       Bob Bergazzi     May 16,1983     Version 6.11
0000     43 ;                Changed the order of .LIB statements.
0000     44 ;
0000     45 ;       05       Richard Brown    12-August-83    Version 6.13
0000     46 ;                Changed $DS_RELBUF so that it will ignore DS$_FRAGBUF
0000     47 ;                errors returned from $CNTREG.  Eliminates problem of
0000     48 ;                RELBUF returning an error if part of the buffer space
0000     49 ;                being deallocated is above the VDS and part is below.
0000     50 ;
0000     51 ;       06       Bob Bergazzi     21-Oct-1983     Version 6.13
0000     52 ;                Added # sign to SS$_NORMAL in edit number 05.
0000     53 ;--
```

```
                     0000     55          .SBTTL  DECLARATIONS
                     0000     56 ;
                     0000     57 ; INCLUDE FILES:
                     0000     58 ;
                     0000     59          .LIBRARY        /$DS/                   ;                                       [4]
                     0000     60          .LIBRARY        /$DIAG/                 ;                                       [4]
                     0000     61
                     0000     62 ;
                     0000     63 ; MACROS:
                     0000     64 ;
                     0000     65
                     0000     66 ;
                     0000     67 ; EQUATED SYMBOLS:
                     0000     68 ;
                     0000     69          $DS_DSADEF
                     0000     70          $DS_DSDEF
                     0000     71          $DS_GETBUF_DEF
                     0000     72          $DS_RELBUF_DEF
                     0000     73
                     0000     74 ;
                     0000     75 ; OWN STORAGE:
                     0000     76 ;
                00000000     77          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
00000008  0000       78 DS$GQ_PHYADR::   .BLKL   2               ; TEMPORARY PHYADR.
00000018  0008       79 DS$GL_BUFCNT::   .BLKL   1               ; KEEPS TRACK OF BUFFER ALLOCATIONS.
00000020  0018       80 Q_RETADR:        .BLKL   2               ; TEMPORARY RETADR.
```

```
                     0020     82          .SBTTL  GET A VERTUAL MEMORY BUFFER.
                 00000000     83          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000     84 ;++
                     0000     85 ; FUNCTIONAL DESCRIPTION:
                     0000     86 ;
                     0000     87 ;       GET A MEMORY BUFFER IS FUNCTIONAL SIMILAR TO STARTLET $EXPREG.
                     0000     88 ;       IT ALLOCATES MEMORY FOR THE PROGRAM TO USE AS BUFFER.
                     0000     89 ;       IN THE REPAIR ENVIRONMENT, THE PHYSICAL MEMORY IS GUARENTEED TO
                     0000     90 ;       BE CONTIGUOUS.
                     0000     91 ;
                     0000     92 ; CALLING SEQUENCE:
                     0000     93 ;
                     0000     94 ;       STANDARD PROCEDURE CALL.
                     0000     95 ;
                     C000     96 ; INPUT PARAMETERS:
                     0000     97 ;
                     0000     98 ;       GETBUF$_PAGCNT(AP) =    NUMBER OF PAGES TO BE ALLOCATED.
                     0000     99 ;       GETBUF$_RETADR(AP) =    ADDRESS OF TWO LONGWORD ARRAY TO RECEIVE
                     0000    100 ;                               VIRTUAL LIMITS.
                     0000    101 ;       GETBUF$_PHYADR(AP) =    ADDRESS OF TWO LONGWORD ARRAY TO RECEIVE
                     0000    102 ;                               PHYSICAL LIMITS.
                     0000    103 ;       GETBUF$_REGION(AP) =    REGION INDICATOR:
                     0000    104 ;                                       0 = PROGRAM (P0) REGION
                     0000    105 ;                                       1 = CONTROL (P1) REGION
                     0000    106 ;                                       2 = SYSTEM REGION.
                     0000    107 ;
                     0000    108 ; IMPLICIT INPUTS:
                     0000    109 ;
                     0000    110 ;       NONE
                     0000    111 ;
                     0000    112 ; OUTPUT PARAMETERS:
                     0000    113 ;
                     0000    114 ;       NONE
                     0000    115 ;
                     0000    116 ; IMPLICIT OUTPUTS:
                     0000    117 ;
                     0000    118 ;       NONE
                     0000    119 ;
                     0000    120 ; COMPLETION CODES:
                     0000    121 ;
                     0000    122 ;       SEE $EXPREG (IN MEMMGT MODULE).
                     0000    123 ;
                     0000    124 ; SIDE EFFECTS:
                     0000    125 ;
                     0000    126 ;       NONE
                     0000    127 ;
                     0000    128 ;--
```

```
                          000C  0000   130 .ENTRY  DSX$GETBUF,^M<R2,R3>            ; Registers to save
                                0002   131
          53   10 AC    02   00   EF  0002   132          EXTZV   #0, #2, GETBUF$_REGION(AP), -
                                0008   133                   R3                   ; REGION AVAILABLE.
                     50   7E   7E  0008   134          MOVAQ   -(SP),R0             ; Point to QUADWORD for start,end
                                000B   135          $EXPREG_S  GETBUF$_PAGCNT(AP), -; PASS ARGUMENTS THRU TO STARLET.
                                000B   136                     (R0), , R3
                          53   DD   000B                PUSHL   R3
                          00   DD   000D                PUSHL   #0
                          60   7F   000F                PUSHAQ  (R0)
                       04 AC   DD   0011                PUSHL   GETBUF$_PAGCNT(AP)
          00000000'GF     04   FB   0014                CALLS   #4,G^SYS$EXPREG
                       0C AC   D5   001B   137          TSTL    GETBUF$_PHYADR(AP)   ; CHECK PHYSICAL ADDRESS ARG.
                          06   13   001E   138          BEQL    20$
           0C BC    0000'CF   7D   C020   139          MOVQ    W^DS$GQ_PHYADR, -    ; COPY ADDRESS ARRAY.
                                0026   140                    @GETBUF$_PHYADR(AP)
                                0026   141
                     51   8E   7D  0026   142 20$:     MOVQ    (SP)+,R1             ; Get first and last address
                     52   51   D1  0029   143          CMPL    R1,R2                ; First really smaller?
                          07   15  002C   144          BLEQ    30$                  ; Branch if so
                          52   DD  002E   145          PUSHL   R2                   ; Save smaller
                     52   51   D0  0030   146          MOVL    R1,R2                ; copy larger to higher register
                          02   BA  0033   147          POPR    #^M<R1>              ; Put smaller in lower register
                                0035   148
                       08 AC   D5  0035   149 30$:     TSTL    GETBUF$_RETADR(AP)   ; CHECK FOR RETURN WANTED.
                          04   13  0038   150          BEQL    40$                  ; SKIP IF NOT.
                08 BC     51   7D  003A   151          MOVQ    R1, @GETBUF$_RETADR(AP) ; STORE BUFFER ADDRESSES.
                                003E   152
                     52   51   C2  003E   153 40$:     SUBL    R1,R2                ; Compute buffer size
          52   52    F7   8F   78  0041   154          ASHL    #-9, R2, R2          ; IN PAGES.
                          52   D6  0046   155          INCL    R2                   ; make it a count
           0008'CF43      52   C0  0048   156          ADDL2   R2,W^DS$GL_BUFCNT[R3]; KEEP TRACK.
                                004E   157
                    FFAF'  30   004E   158          BSBW    KB_CHECK             ; ALLOW ^C.
                           04   0051   159          RET
```

```
0052   161          .SBTTL  RELEASE A MEMORY BUFFER.
0052   162 ;++
0052   163 ; FUNCTIONAL DESCRIPTION:
0052   164 ;
0052   165 ;
0052   166 ; CALLING SEQUENCE:
0052   167 ;
0052   168 ;       STANDARD PROCEDURE CALL.
0052   169 ;
0052   170 ; INPUT PARAMETERS:
0052   171 ;
0052   172 ;       RELBUF$_PAGCNT(AP) =    NUMBER OF PAGES TO BE ALLOCATED.
0052   173 ;       RELBUF$_RETADR(AP) =    ADDRESS OF TWO LONGWORD ARRAY TO RECEIVE
0052   174 ;                               VIRTUAL LIMITS.
C052   175 ;       RELBUF$_REGION(AP) =    REGION INDICATOR:
0052   176 ;                                       0 = PROGRAM (P0) REGION
0052   177 ;                                       1 = CONTROL (P1) REGION
0052   178 ;                                       2 = SYSTEM REGION.
00>2   179 ;
0052   180 ; IMPLICIT INPUTS:
0052   181 ;
0052   182 ;       NONE
0052   183 ;
0052   184 ; OUTPUT PARAMETERS:
0052   185 ;
0052   186 ;       NONE
0052   187 ;
0052   188 ; IMPLICIT OUTPUTS:
0052   189 ;
0052   190 ;       NONE
0052   191 ;
0052   192 ; COMPLETION CODES:
0052   193 ;
0052   194 ;       SEE $CNTREG (IN MEMMGT MODULE).
0052   195 ;
0052   196 ; SIDE EFFECTS:
0052   197 ;
0052   198 ;       NONE
0052   199 ;
0052   200 ;--
```

```
                          0004  0052   202 .ENTRY  DSX$RELBUF,^M<R2>                ; ENTRY MASK.
                                0054   203
        52   0C AC   02   00   EF  0054   204        EXTZV   #0,#2,RELBUF$_REGION(AP), R2     ; REGION AVAILABLE.
                51   04 AC   D0  005A   205        MOVL    RELBUF$_PAGCNT(AP), R1           ; USE CALLER PAGE COUNT.
              51   0008'CF42   D1  005E   206        CMPL    W^DS$GL_BUFCNT[R2], R1           ; CHECK FOR TOO LARGE PAGCNT.
                          06   18  0064   207        BGEQ    20$                              ; SKIP IF NOT TOO LARGE.
              51   0008'CF42   D0  0066   208        MOVL    W^DS$GL_BUFCNT[R2], R1           ; GET MAXIMUM PAGE COUNT.
                                006C   209
                     50   7E   7E  006C   210 20$:    MOVAQ   -(SP),R0                         ; Address of desc
                                006F   211        $CNTREG_S  R1, (R0), , R2                ; CONTRACT REGION.
                          52   DD  006F              PUSHL   R2
                          00   DD  0071              PUSHL   #0
                          60   7F  0073              PUSHAQ  (R0)
                          51   DD  0075              PUSHL   R1
        00000000'GF       04   FB  C077              CALLS   #4,G^SYS$CNTREG
        00660080 8F       50   D1  007E   212        CMPL    R0,#DS$_FRAGBUF                  ; IF FRAGBUF error code [5]
                          07   12  0085   213        BNEQ    25$                              ; THEN                  [5]
        50   00000000'8F       D0  0087   214        MOVL    #SS$_NORMAL,R0                   ;    ignore it.         [06]
                                008E   215 25$:                                             ;                       [5]
                    08 AC   D5  008E   216        TSTL    RELBUF$_RETADR(AP)               ; CHECK FOR RETURN WANTED.
                          04   13  0091   217        BEQL    30$                              ; SKIP IF NOT.
                 08 BC   6E   7D  0093   218        MOVQ    (SP), @RELBUF$_RETADR(AP)        ; STORE BUFFER ADDRESSES.
                                0097   219
              51   8E   8E   C3  0097   220 30$:    SUBL3   (SP)+,(SP)+, R1                  ; COMPUTE BUFFER SIZE.
                          03   18  009B   221        BGEQ    35$                              ; Branch if positive
                    51   51   CE  009D   222        MNEGL   R1,R1                            ; Make it positive
                                00A0   223
        51   51   F7 8F   78  00A0   224 35$:    ASHL    #-9, R1, R1                      ; IN PAGES.
                          51   D6  00A5   225        INCL    R1                               ; MAKE INTO COUNT.
         0008'CF42        51   C2  00A7   226        SUBL2   R1, W^DS$GL_BUFCNT[R2]           ; KEEP TRACK.
                    0B 50   E9  00AD   227        BLBC    R0, 40$                          ; SKIP IF ALREADY ERROR.
              04 AC   51   D1  00B0   228        CMPL    R1, RELBUF$_PAGCNT(AP)           ; CHECK IF TOTAL SUCCESS.
                          05   13  00B4   229        BEQL    40$                              ; SKIP IF OK.
           50   0000'8F   3C  00B6   230        MOVZWL  #SS$_PAGOWNVIO, R0               ; ERROR CODE.
                                00BB   231
                    FF42'   30  00BB   232 40$:    BSBW    KB_CHECK                         ; ALLOW ^C.
                          04   00BE   233        RET
                                00BF   234
                                00BF   235        .END
```

```
$$ARGS                          =  00000003     D
$$T1                            =  00000000     D
BIT...                          =  00660130     D
DS$GL_BUFCNT                       00000008 RG  D  02
DS$GQ_PHYADR                       00000000 RG  D  02
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_WARNING                    =  00000000     D
DS$_ARITH                       =  006600D0     D
DS$_ASBE                        =  00660118     D
DS$_BADLINK                     =  006600F0     D
DS$_BADTYPE                     =  006600E8     D
DS$_BIIC                        =  00660120     D
DS$_CHME                        =  006600A8     D
DS$_CHMK                        =  006600E0     D
DS$_DEVNAME                     =  00660108     D
DS$_ERROR                       =  00660002     D
DS$_FHWE                        =  00660068     D
DS$_FRAGBUF                     =  00660080     D
DS$_ICBUSY                      =  006600C8     D
DS$_ICERR                       =  006600C0     D
DS$_IHWE                        =  00660060     D
DS$_ILLCHAR                     =  00660018     D
DS$_ILLPAGCNT                   =  00660078     D
DS$_ILLUNIT                     =  00660100     D
DS$_INSFMEM                     =  00660050     D
DS$_IPL2HI                      =  006600B8     D
DS$_IVADDR                      =  00660040     D
DS$_IVVECT                      =  00660038     D
DS$_KRNLSTK                     =  00660090     D
DS$_LOGIC                       =  00660070     D
DS$_MCHK                        =  00660088     D
DS$_MMOFF                       =  00660058     D
DS$_NEEDUNIT                    =  006600F8     D
DS$_NODE                        =  00660128     D
DS$_NOPCS                       =  00660110     D
DS$_NORMAL                      =  00660001     D
DS$_NOSUPPORT                   =  006600B1     D
DS$_NOTDON                      =  00660030     D
DS$_NOTIMP                      =  006600B0     D
DS$_NULLSTR                     =  00660010     D
DS$_OVERFLOW                    =  00660008     D
DS$_POWER                       =  00660098     D
DS$_PROGERR                     =  00660020     D
DS$_SEVERE                      =  00660004     D
DS$_TRANSL                      =  006600A0     D
DS$_TRUNCATE                    =  00660028     D
DS$_UNEXPINT                    =  006600D8     D
DS$_VASFULL                     =  00660048     D
DS$_WARNING                     =  00660000     D
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSX$GETBUF                         00000000 RG  D  03
DSX$RELBUF                         00000052 RG  D  03
GETBUF$_NARGS                   =  00000004     D
GETBUF$_PAGCNT                  =  00000004     D
GETBUF$_PHYADR                  =  0000000C     D
GETBUF$_REGION                  =  00000010     D
GETBUF$_RETADR                  =  00000008     D
KB_CHECK                           ********   X   03
Q_RETADR                           00000018 R  D  02
RELBUF$_NARGS                   =  00000003     D
RELBUF$_PAGCNT                  =  00000004     D
RELBUF$_REGION                  =  0000000C     D
RELBUF$_RETADR                  =  00000008     D
SIZ...                          =  00000001     D
SS$_NORMAL                         ********   X   03
SS$_PAGOWNVIO                      ********   X   03
SYS$CNTREG                         ********  GX   03
SYS$EXPREG                         ********  GX   03
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000020 (32.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	000000BF (191.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000003	72 (1)	71 (1) 72 (1)
$$T1	=00000000	211 (5)	136 (3) 211 (5) 71 (1) 72 (1)
BIT...	=00660130	70 (1)	70 (1)
DS$GL_BUFCNT	00000008-R	79 (1)	#-156 (3) #-206 (5) #-208 (5) #-226 (5)
DS$GQ_PHYADR	00000000-R	78 (1)	#-139 (3)
DS$K_ERROR	=00000002	70 (1)	
DS$K_NORMAL	=00000001	70 (1)	
DS$K_SEVERE	=00000004	70 (1)	
DS$K_SUBSYS	=00000066	70 (1)	70 (1)
DS$K_WARNING	=00000000	70 (1)	
DS$_ARITH	=006600D0	70 (1)	
DS$_ASBE	=00660118	70 (1)	
DS$_BADLINK	=006600F0	70 (1)	
DS$_BADTYPE	=006600E8	70 (1)	
DS$_BIIC	=00660120	70 (1)	
DS$_CHME	=006600A8	70 (1)	
DS$_CHMK	=006600E0	70 (1)	
DS$_DEVNAME	=00660108	70 (1)	
DS$_ERROR	=00660002	70 (1)	
DS$_FHWE	=00660068	70 (1)	
DS$_FRAGBUF	=00660080	70 (1)	#-212 (5)
DS$_ICBUSY	=006600C8	70 (1)	
DS$_ICERR	=006600C0	70 (1)	
DS$_IHWE	=00660060	70 (1)	
DS$_ILLCHAR	=00660018	70 (1)	
DS$_ILLPAGCNT	=00660078	70 (1)	
DS$_ILLUNIT	=00660100	70 (1)	
DS$_INSFMEM	=00660050	70 (1)	
DS$_IPL2HI	=006600B8	70 (1)	
DS$_IVADDR	=00660040	70 (1)	
DS$_IVVECT	=00660038	70 (1)	
DS$_KRNLSTK	=00660090	70 (1)	
DS$_LOGIC	=00660070	70 (1)	
DS$_MCHK	=00660088	70 (1)	
DS$_MMOFF	=00660058	70 (1)	
DS$_NEEDUNIT	=006600F8	70 (1)	
DS$_NODE	=00660128	70 (1)	
DS$_NOPCS	=00660110	70 (1)	
DS$_NORMAL	=00660001	70 (1)	
DS$_NOSUPPORT	=006600B1	70 (1)	
DS$_NOTDON	=00660030	70 (1)	
DS$_NOTIMP	=006600B0	70 (1)	70 (1)
DS$_NULLSTR	=00660010	70 (1)	
DS$_OVERFLOW	=00660008	70 (1)	
DS$_POWER	=00660098	70 (1)	
DS$_PROGERR	=00660020	70 (1)	
DS$_SEVERE	=00660004	70 (1)	
DS$_TRANSL	=006600A0	70 (1)	
DS$_TRUNCATE	=00660028	70 (1)	
DS$_UNEXPINT	=006600D8	70 (1)	

```
DS$_VASFULL                     =00660048      70      (1)
DS$_WARNING                     =00660000      70      (1)       70      (1)
DSX$GETBUF                       00000000-R    130     (3)
DSX$RELBUF                       00000052-R    202     (5)
GETBUF$_NARGS                   =00000004      71      (1)
GETBUF$_PAGCNT                  =00000004      71      (1)     #-136     (3)
GETBUF$_PHYADR                  =0000000C      71      (1)     #-137     (3)     #-140     (3)
GETBUF$_REGION                  =00000010      71      (1)       132     (3)
GETBUF$_RETADR                  =00000008      71      (1)     #-149     (3)     #-151     (3)
KB_CHECK                         00000000-XR                   #-158     (3)     #-232     (5)
Q_RETADR                         00000018-R    80      (1)
RELBUF$_NARGS                   =00000003      72      (1)
RELBUF$_PAGCNT                  =00000004      72      (1)     #-205     (5)     #-228     (5)
RELBUF$_REGION                  =0000000C      72      (1)       204     (5)
RELBUF$_RETADR                  =00000008      72      (1)     #-216     (5)     #-218     (5)
SS$_NORMAL                       00000000-XR                   #-214     (5)
SS$_PAGOWNVIO                    00000000-XR                   #-230     (5)
SYS$CNTREG                       00000000-XR                     211     (5)
SYS$EXPREG                       00000000-XR                     136     (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ASNPUSH	1	136 (3)	136 (3) 211 (5)
$CNTREG_S	1	211 (5)	211 (5)
$DEF	1	70 (1)	
$DEFINI	1	69 (1)	69 (1)
$DS_DSADEF	5	69 (1)	69 (1)
$DS_DSDEF	2	70 (1)	70 (1)
$DS_GETBUF_DEF	1	71 (1)	71 (1)
$DS_RELBUF_DEF	1	72 (1)	72 (1)
$EQU	1	70 (1)	70 (1)
$EQULS1	1	70 (1)	70 (1)
$EQULST	1	70 (1)	70 (1)
$EXPREG_S	1	135 (3)	135 (3)
$GBLINI	2		70 (1)
$OFFDEF	1	71 (1)	71 (1) 72 (1)
$PUSHADR	1	136 (3)	136 (3) 211 (5)
$VIELD1	1	70 (1)	

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.12	00:00:00.49
Command processing	140	00:00:00.72	00:00:02.21
Pass 1	338	00:00:04.50	00:00:08.71
Symbol table sort	0	00:00:00.15	00:00:00.16
Pass 2	74	00:00:00.88	00:00:00.95
Symbol table output	11	00:00:00.06	00:00:00.08
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	17	00:00:00.30	00.00:00.30
Assembler run totals	620	00:00:06.77	00:00:12.94

The working set limit was 1000 pages.
45664 bytes (90 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 140 non-local and 8 local symbols.
235 source lines were read in Pass 1, producing 0 object records in Pass 2.
38 pages of virtual memory were used to define 16 macros.

```
                                        +----------------------------+
                                        ! Macro library statistics !
                                        +----------------------------+

Macro library name                       Macros defined
------------------                       --------------
DRB1:[DS.WORK]DIAG.MLB;955                        4
DRB1:[DS.WORK]DS.MLB;218                          0
DMA1:[SYSO.SYSMAINT]DS.MLB;218                    0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953                  0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1                     0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2                10
TOTALS (all libraries)                           14

249 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) BUFFER/UPDA=(BUFFER.UPD,BUFFER.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

able of contents

(1) 64 CANCEL I/O ON CHANNEL

```
                        0000     .1          .TITLE  CANCEL  *** CANCEL cancel QIO
                        0000     .2          .IDENT  /05-04/
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 4-AUG-77
                        0000     27 ;
                        0000     28 ; SYSTEM SERVICE CANCEL I/O ON CHANNEL
                        0000     29 ;
                        0000     30 ; MODIFIED BY:
                        0000     31 ;
                        0000     .1 ;                N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
                        0000     .2 ;        01      DIAGNOSTIC SUPERVISOR INTEGRATION
                        0000     .3 ;
                        0000     .4 ;                Dave Butenhof   13-may-1980, Version 5.4
                        0000     .5 ;        02      Modify VMS V2.0 source for Supervisor environment
                        0000     .6 ;                Roger Riggs, 23-July-1980, Version 5.5
                        0000     .7 ;        03      Changed word offset to long
                        0000     .8 ;
                        0000     .9 ;        04      M. Baggett      13-May-1983     Version 6.11
                        0000    .10 ;                Fixed trucation error.
                        0000    .11 ;
                        0000     32 ;        V0002   ACG0047         Andrew C. Goldstein,    16-Aug-1979  18:37
                        0000     33 ;                Add access rights block, protection interface changes
                        0000     34 ;
                        0000     35 ;**
                        0000     36 ;
                        0000     37 ; MACRO LIBRARY CALLS
                        0000     38 ;
                        0000     39
                        0000     40          $CADEF                          ;DEFINE CONDIATIONAL ASSEMBLY PARAMETERS
                        0000     41          $CCBDEF                         ;DEFINE CCB OFFSETS
                        0000     42          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     43          $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000     44          $DEVDEF                         ;DEFINE DEVICE CHARACTERISTIC BITS
                        0000     45          $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                        0000     46          $IODEF                          ;DEFINE I/O FUNCTION CODES
```

```
                        0000     47          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000     48          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000     49          $PCBDEF                         ;DEFINE PCB OFFSETS
                        0000     50          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     51          $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000     52          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000     53          $VCBDEF                         ;DEFINE VCB OFFSETS
                        0000     54          $WCBDEF                         ;DEFINE WCB OFFSETS
                        0000     55
                        0000     56 ;
                        0000     57 ; LOCAL SYMBOLS
                        0000     58 ;
                        0000     59 ; ARGUMENT LIST OFFSET DEFINITIONS
                        0000     60 ;
                        C000     61
                00000004  0000     62 CHAN=4                                  ;I/O CHANNEL NUMBER
                        00000000     .1          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                        0000     .2          MODNAM  CANCEL
```

```
                                0007    64          .SBTTL  CANCEL I/O ON CHANNEL
                                0007    65 ;+
                                0007    66 ; EXE$CANCEL - CANCEL I/O ON CHANNEL
                                0007    67 ;
                                0007    68 ; THIS SERVICE CANCELS ALL I/O ISSUED TO A DEVICE FORM THE SPECIFIED CHANNEL.
                                0007    69 ;
                                0007    70 ; INPUTS:
                                0007    71 ;
                                0007    72 ;       CHAN(AP) = NUMBER OF THE I/O CHANNEL TO CANCEL I/O FOR.
                                0007    73 ;
                                0007    74 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                0007    75 ;
                                0007    76 ; OUTPUTS:
                                0007    77 ;
                                0007    78 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CANCEL I/O.
                                0007    79 ;
                                0007    80 ;               SS$_EXQUOTA - DIRECT I/O QUOTA EXCEEDED WHILE TRYING TO
                                0007    81 ;                     CANCEL FILE I/O.
                                0007    82 ;
                                0007    83 ;               SS$_INSFMEM - INSUFFICIENT MEMORY AVAILABLE TO ALLOCATE I/O
                                0007    84 ;                     PACKET.
                                0007    85 ;
                                0007    86 ;               SS$_IVCHAN - INVALID CHANNEL NUMBER SPECIFIED.
                                0007    87 ;
                                0007    88 ;               SS$_NOPRIV - SPECIFIED CHANNEL IS NOT ASSIGNED TO A DEVICE
                                0007    89 ;                     OR THE CALLER DOES NOT HAVE SUFFICIENT PRIVILEGE TO
                                0007    90 ;                     ACCESS THE CHANNEL.
                                0007    91 ;
                                0007    92 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0007    93 ;
                                0007    94 ;               SS$_NORMAL - NORMAL COMPLETION.
                                0007    95 ;-
                                0007    96
                          00FC  0007    97          .ENTRY  EXE$CANCEL,^M<R2,R3,R4,R5,R6,R7>
00000000'EF   9E  0009    .1          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
        54        000F
 50   04 AC   3C  0010    98          MOVZWL  CHAN(AP),R0             ;GET I/O CHANNEL NUMBER
00000000'EF   16  0014    99          JSB     IOC$VERIFYCHAN          ;VERIFY I/O CHANNEL NUMBER
      60 50   E9  001A   100          BLBC    R0,50$                  ;IF LBC INVALID CHANNEL
      57 52   D0  001D   101          MOVL    R2,R7                   ;SAVE CHANNEL INDEX
      56 51   D0  0020   102          MOVL    R1,R6                   ;SAVE ADDRESS OF CCB
      55 66   D0  0023   103          MOVL    CCB$L_UCB(R6),R5        ;GET ASSIGNED DEVICE UCB ADDRESS
         7E   DC  0026   104 10$:     MOVPSL  -(SP)                   ;SAVE CURRENT PROCESSOR STATUS
              0028   105          SETIPL  #IPL$_ASTDEL            ;RAISE TO AST DELIVERY LEVEL
      0A A6   B5  002B   106          TSTW    CCB$W_IOC(R6)           ;ANY I/O OUTSTANDING?
      51      13  002E   107          BEQL    60$                     ;IF EQL NO
 53   40 A5   9E  0030   108          MOVAB   UCB$L_IOQFL(R5),R3      ;GET ADDRESS OF I/O QUEUE LISTHEAD
    52   53   D0  0034   109          MOVL    R3,R2                   ;COPY ADDRESS OF I/O QUEUE LISTHEAD
              0037   110          SETIPL  UCB$B_FIPL(R5)          ;RAISE TO DRIVER FORK IPL
    52   62   D0  003B   111 20$:     MOVL    IRP$L_IOQFL(R2),R2      ;GET ADDRESS OF NEXT I/O PACKET IN QUEUE
    53   52   D1  003E   112          CMPL    R2,R3                   ;END OF QUEUE?
         3E   13  0041   113          BEQL    60$                     ;IF EQL YES
F3 2A A2   04   E0  0043   114          BBS     #IRP$V_VIRTUAL,IRP$W_STS(R2),20$ ;IF SET, VIRTUAL I/O REQUEST
60 A4   0C A2   D1  0048   115          CMPL    IRP$L_PID(R2),PCB$L_PID(R4) ;PROCESS ID MATCH?
         EC   12  004D   116          BNEQ    20$                     ;IF NEQ NO
   28 A2   57   B1  004F   117          CMPW    R7,IRP$W_CHAN(R2)       ;I/O CHANNEL NUMBER MATCH?
         E6   12  0053   118          BNEQ    20$                     ;IF NEQ NO
```

```
                    52    04 A2    D0  0055   119          MOVL    IRP$L_IOQBL(R2),R2      ;GET BACKWARD LINK OF CURRENT ENTRY
                    51    00 B2    0F  0059   120          REMQUE  @IRP$L_IOQFL(R2),R1     ;REMOVE I/O PACKET FROM QUEUE
                 04 2A A1    00    E1  005D   121          BBC     #IRP$V_BUFIO,IRP$W_STS(R1),30$ ;IF CLR, DIRECT I/O REQUEST
                    2A A1    02    AA  0062   122          BICW    #IRP$M_FUNC,IRP$W_STS(R1) ;CLEAR BUFFERED READ
                       0000'8F    3C  0066   123 30$:      MOVZWL  #SS$_CANCEL,IRP$L_MEDIA(R1) ;SET COMPLETION STATUS
                          34 A1       006A
                             61    0E  006C    .1          INSQUE  IRP$L_IOQFL(R1),@L^IOC$GL_PSBL ;INSERT PACKET IN POST PROCESS QUEUE
                 00000000'FF          006E
-1                           C6    12  0073   125          BNEQ    20$                     ;IF NEQ NOT FIRST ENTRY IN QUEUE
                                     0075   126          SOFTINT #IPL$_IOPOST            ;INITIATE SOFTWARE INTERRUPT
                             C1    11  0078   127          BRB     20$                     ;
                       50    00'   3C  007A   128 40$:     MOVZWL  S^#SS$_NORMAL,R0        ;SET NORMAL COMPLETION STATUS
                                     007D   129 50$:     SETIPL  #0                      ;ALLOW INTERRUPTS
                                04   0080   130          RET                             ;
                                     C081   131 60$:     SETIPL  UCB$B_FIPL(R5)          ;RAISE TO DRIVER FORK LEVEL
                    50    24 A5    D0  0085   132          MOVL    UCB$L_DDB(R5),R0        ;GET ADDRESS OF DDB
                    50    0C A0    D0  0089   133          MOVL    DDB$L_DDT(R0),R0        ;GET ADDRESS OF DDT
                    53    4C A5    D0  008D   134          MOVL    UCB$L_IRP(R5),R3        ;GET CURRENT I/O PACKET ADDRESS
                       52    57    D0  0091   135          MOVL    R7,R2                   ;SET CHANNEL INDEX
                    51    0C A0    D0  0094    .1          MOVL    DDT$L_CANCEL(R0),R1     ;GET ADDRESS OF CANCEL I/O ENTRY POINT
                             E3    13  0098    .2          BEQL    50$                     ;IF EQL NONE
                 03 51       10    E0  009A    .3          BBS     #16,R1,70$              ;++++ BRANCH IF SUPERVISOR ABSOLUTE
                    51       50    C0  009E    .4          ADDL    R0,R1                   ;++++ OFFSET BY DDT BASE
-3                           61    16  00A1   139 70$:     JSB     (R1)                    ;CALL CANCEL I/O ROUTINE
                    50    0A A6    3C  00A3   140          MOVZWL  CCB$W_IOC(R6),R0        ;GET OUTSTANDING I/O COUNT
                    50    04 A6    C8  00A7   141          BISL    CCB$L_WIND(R6),R0       ;OUTSTANDING I/O OR FILE ACTIVITY?
                             CD    13  00AB   142          BEQL    40$                     ;IF EQL NO
                 C8 34 A5    13    E1  00AD   143          BBC     #DEV$V_MNT,UCB$L_DEVCHAR(R5),40$ ;IF CLR, DEVICE DISMOUNTED
                 C3 34 A5    18    E0  00B2   144          BBS     #DEV$V_FOR,UCB$L_DEVCHAR(R5),40$ ;IF SET, MOUNTED FOREIGN
                    51    5C 8F    9A  00B7    .1          MOVZBL  #IRP$C_LENGTH,R1        ;SET LENGTH OF I/O PACKET
                   00000000'EF    16  00BB    .2          JSB     EXE$ALONONPAGED         ;ALLOCATE I/O PACKET                    [04]
-10                       0B 50    E8  00C1   155          BLBS    R0,100$                 ;IF LBS SUCCESSFUL ALLOCATION
                 51    0000'8F    3C  00C4   156          MOVZWL  #SS$_INSFMEM,R1         ;SET INSUFFICIENT MEMORY STATUS
                       50    03    3C  00C9   157          MOVZWL  #RSN$_NPDYNMEM,R0       ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
                          FF57       31  00CC    .1 90$:     BRW     10$
-6                           3A A4    B7  00CF   164 100$:    DECW    PCB$W_BIOCNT(R4)        ;UPDATE BUFFERED I/O COUNT
                             0A A6    B6  00D2   165          INCW    CCB$W_IOC(R6)           ;INCREMENT I/O COUNT
                       53    52    D0  00D5   166          MOVL    R2,R3                   ;COPY ADDRESS OF I/O PACKET
                       52    08    C0  00D8   167          ADDL    #IRP$W_SIZE,R2          ;CALCULATE ADDRESS OF PACKET SIZE
                    82    5C 8F    9B  00DB   168          MOVZBW  #IRP$C_LENGTH,(R2)+     ;INSERT SIZE OF I/O REQUEST PACKET
                       82    0A    9B  00DF   169          MOVZBW  #DYN$C_IRP,(R2)+        ;INSERT DATA STRUCTURE TYPE AND ZERO MODE
                    82    60 A4    D0  00E2   170          MOVL    PCB$L_PID(R4),(R2)+     ;INSERT CURRENT PROCESS ID
                             82    7C  00E6   171          CLRQ    (R2)+                   ;CLEAR AST ADDRESS AND PARAMETER
                    82    04 A6    D0  00E8   172          MOVL    CCB$L_WIND(R6),(R2)+    ;INSERT ADDRESS OF WINDOW
                       82    55    D0  00EC   173          MOVL    R5,(R2)+                ;INSERT DEVICE UCB ADDRESS
                       82    38    B0  00EF   174          MOVW    #IO$_ACPCONTROL,(R2)+   ;INSERT I/O FUNCTION CODE
                       82    1F    90  00F2   175          MOVB    #31,(R2)+               ;SET EVENT FLAG NUMBER
                       82    00    90  00F5    .1          MOVB    #0,(R2)+                ;INSERT PROCESS BASE PRIORITY
                             82    D4  00F8    .2          CLRL    (R2)+                   ;CLEAR I/O STATUS BLOCK ADDRESS
                       82    57    3C  00FA    .3          MOVZWL  R7,(R2)+                ;INSERT CHANNEL NUMBER AND CLEAR STATUS
                             82    7C  00FD    .4          CLRQ    (R2)+                   ;CLEAR BUFFER PARAMETERS
-6                  50    30 A5    D0  00FF   182          MOVL    UCB$L_VCB(R5),R0        ;GET ADDRESS OF VCB
                       0C A0    B6  0103   183          INCW    VCB$W_TRANS(R0)         ;UPDATE TRANSACTION COUNT
                                    0106   184
-6                                  0106   191
                          FEF7'    31  0106   192          BRW     EXE$QIOACPPKT           ;QUEUE ACP PACKET
                                    0109   193
```

```
                                0109    194          .END
```

```
$ER                 =  00000001     D
$MODULE                00000000 R   D   02
CCB$B_AMOD             00000009     D
CCB$B_STS              00000008     D
CCB$C_LENGTH           00000010     D
CCB$K_LENGTH           00000010     D
CCB$L_DIRP             000C000C     D
CCB$L_UCB              00000000     D
CCB$L_WIND             00000004     D
CCB$W_IOC              0000000A     D
CHAN                =  00000004     D
DDB$B_ACPCLASS         00000013     D
DDB$B_TYPE             0000000A     D
DDB$C_LENGTH           00000034     D
DDB$K_LENGTH           00000034     D
DDB$L_ACPD             00000010     D
DDB$L_DDT              0000000C     D
DDB$L_LINK             00000000     D
DDB$L_UCB              00000004     D
DDB$T_DRVNAME          00000024     D
DDB$T_NAME             00000014     D
DDB$W_SIZE             00000008     D
DDT$L_ALTSTART         0000001C     D
DDT$L_CANCEL           0000000C     D
DDT$L_FDT              00000008     D
DDT$L_REGDUMP          00000010     D
DDT$L_START            00000000     D
DDT$L_UNITINIT         00000018     D
DDT$L_UNSOLINT         00000004     D
DDT$W_DIAGBUF          00000014     D
DDT$W_ERRORBUF         00000016     D
DEV$V_FOR           =  00000018     D
DEV$V_MNT           =  00000013     D
DS$AX_SOFTPCB          ********     X   02
DYN$C_IRP           =  0000000A     D
EXE$ALONONPAGED        ********     X   02
EXE$CANCEL             00000007 RG  D   02
EXE$QIOACPPKT          ********     X   02
IO$_ACPCONTROL      =  00000038     D
IOC$GL_PSBL            ********     X   02
IOC$VERIFYCHAN         ********     X   02
IPL$_ASTDEL         =  00000002     D
IPL$_IOPOST         =  00000004     D
IRP$B_CARCON           00000038     D
IRP$B_EFN              00000022     D
IRP$B_PRI              00000023     D
IRP$B_RMOD             0000000B     D
IRP$B_TYPE             0000000A     D
IRP$C_LENGTH           0000005C     D
IRP$K_LENGTH           0000005C     D
IRP$L_ARB              00000050     D
IRP$L_AST              00000010     D
IRP$L_ASTPRM           00000014     D
IRP$L_DIAGBUF          00000044     D
IRP$L_EXTEND           0000004C     D
IRP$L_IOQBL            00000004     D
IRP$L_IOQFL            00000000     D
IRP$L_IOSB             00000024     D
IRP$L_IOST1            00000034     D
IRP$L_IOST2            00000038     D
IRP$L_MEDIA            00000034     D
IRP$L_PID              0000000C     D
IRP$L_SEGVBN           00000040     D
IRP$L_SEQNUM           00000048     D
IRP$L_SVAPTE           0000002C     D
IRP$L_TT_TERM          00000038     D
IRP$L_UCB              0000001C     D
IRP$L_WIND             00000018     D
IRP$M_FUNC          =  00000002     D
IRP$Q_NT_PRVMSK        0000003C     D
IRP$V_BUFIO         =  00000000     D
IRP$V_VIRTUAL       =  00000004     D
IRP$W_ABCNT            0000003C     D
IRP$W_BCNT             00000032     D
IRP$W_BOFF             00000030     D
IRP$W_CHAN             00000028     D
IRP$W_FUNC             00000020     D
IRP$W_OBCNT            0000003E     D
IRP$W_SIZE             00000008     D
IRP$W_STS              0000002A     D
IRP$W_TT_PRMPT         0000003C     D
PCB$B_ASTACT           0000000C     D
PCB$B_ASTEN            0000000D     D
PCB$B_PRI              0000000B     D
PCB$B_PRIB             0000002F     D
PCB$B_TYPE             0000000A     D
PCB$B_WEFC             0000002E     D
PCB$C_LENGTH           0000008C     D
PCB$K_LENGTH           0000008C     D
PCB$L_ARB              00000084     D
PCB$L_ASTQBL           00000014     D
PCB$L_ASTQFL           00000010     D
PCB$L_EFC2P            00000058     D
PCB$L_EFC3P            0000005C     D
PCB$L_EFCS             00000050     D
PCB$L_EFCU             00000054     D
PCB$L_EFWM             0000004C     D
PCB$L_JIB              00000078     D
PCB$L_OWNER            0000001C     D
PCB$L_PHD              00000064     D
PCB$L_PHYPCB           00000018     D
PCB$L_PID              00000060     D
PCB$L_PQB              0000004C     D
PCB$L_SQBL             00000004     D
PCB$L_SQFL             00000000     D
PCB$L_STS              00000024     D
PCB$L_UIC              00000088     D
PCB$L_WSSWP            00000020     D
PCB$L_WTIME            00000028     D
PCB$Q_PRIV             0000007C     D
PCB$T_LNAME            00000068     D
PCB$T_TERMINAL         00000044     D
PCB$W_APTCNT           00000030     D
PCB$W_ASTCNT           00000038     D
PCB$W_BIOCNT           0000003A     D
PCB$W_BIOLM            0000003C     D
PCB$W_DIOCNT           0000003E     D
PCB$W_DIOLM            00000040     D
PCB$W_GPGCNT           00000034     D
PCB$W_GRP              0000008A     D
PCB$W_MEM              00000088     D
PCB$W_MTXCNT           0000000E     D
PCB$W_PPGCNT           00000036     D
PCB$W_PRCCNT           00000042     D
PCB$W_SIZE             00000008     D
PCB$W_STATE            0000002C     D
PCB$W_TMBU             00000032     D
PR$_IPL             =  00000012     D
PR$_SIRR            =  00000014     D
RSN$_NPDYNMEM       =  00000003     D
SIZ...              =  00000001     D
SS$_CANCEL             ********     X   02
SS$_INSFMEM            ********     X   02
SS$_NORMAL             ********     X   02
UCB$B_AMOD             00000053     D
UCB$B_CEX              00000077     D
UCB$B_CM1              0000004A     D
UCB$B_CM2              0000004B     D
UCB$B_DEVCLASS         00000038     D
UCB$B_DEVTYPE          00000039     D
UCB$B_DIPL             00000052     D
UCB$B_DX_SCTCNT        000000A6     D
UCB$B_ERTCNT           00000070     D
UCB$B_ERTMAX           00000071     D
UCB$B_FEX              00000076     D
UCB$B_FIPL             0000000B     D
UCB$B_LOCSRV           0000003C     D
UCB$B_OFFNDX           00000094     D
UCB$B_OFFRTC           00000095     D
UCB$B_REMSRV           0000003D     D
UCB$B_SECTORS          0000003C     D
UCB$B_SLAVE            00000074     D
UCB$B_SPR              00000075     D
UCB$B_STATE            00000052     D
UCB$B_TRACKS           0000003D     D
UCB$B_TT_CRFILL        0000009D     D
UCB$B_TT_DECRF         000000A1     D
UCB$B_TT_DELFF         000000A2     D
UCB$B_TT_DESPEE        000000A0     D
UCB$B_TT_DETYPE        000000A4     D
UCB$B_TT_LFFILL        0000009E     D
UCB$B_TT_SPEED         0000009C     D
UCB$B_TYPE             0000000A     D
UCB$B_VERTSZ           0000003F     D
UCB$C_LENGTH           00000074     D
UCB$C_MB_LENGTH        00000090     D
UCB$C_TT_LENGTH        000000BC     D
UCB$K_LENGTH           00000074     D
UCB$K_MB_LENGTH        00000090     D
UCB$K_TT_LENGTH        000000BC     D
UCB$L_AMB              00000054     D
```

UCB$L_ASTQBL	00000010	D	UCB$W_DEVSTS	0000005A	D	VCB$L_RVT	00000020	D
UCB$L_ASTQFL	0000000C	D	UCB$W_DIRSEQ	00000088	D	VCB$L_SBMAPLBN	00000034	D
UCB$L_CPID	0000005C	D	UCB$W_DSTADDR	00000018	D	VCB$L_START_FID	00000028	D
UCB$L_CRB	00000020	D	UCB$W_DX_BCR	000000A4	D	VCB$L_ST_RECORD	00000030	D
UCB$L_DDB	00000024	D	UCB$W_ECT	00000090	D	VCB$L_USRLBLAST	00000044	D
UCB$L_DEVCHAR	00000034	D	UCB$W_EC2	00000092	D	VCB$L_VPBL	00000040	D
UCB$L_DEVDEPEND	0000003C	D	UCB$W_ERRCNT	00000072	D	VCB$L_VPFL	0000003C	D
UCB$L_DPC	00000080	D	UCB$W_FUNC	0000007E	D	VCB$L_WCB	00000038	D
UCB$L_DUETIM	0000005C	D	UCB$W_MB_SEED	00000000	D	VCB$T_QNAME	0000000C	D
UCB$L_DX_BFPNT	0000009C	D	UCB$W_MSGCNT	00000016	D	VCB$T_VOLNAME	00000014	D
UCB$L_DX_BUF	00000098	D	UCB$W_MSGMAX	00000014	D	VCB$W_CLUSTER	0000003C	D
UCB$L_DX_RXDB	000000A0	D	UCB$W_NT_CHAN	0000007C	D	VCB$W_CUR_NUM	00000024	D
UCB$L_EMB	00000078	D	UCB$W_OFFSET	0000008A	D	VCB$W_CUR_SEQ	00000026	D
UCB$L_FIRST	00000014	D	UCB$W_REFC	00000050	D	VCB$W_EXTEND	0000003E	D
UCB$L_FPC	0000000C	D	UCB$W_SIZE	00000008	D	VCB$W_FILEPROT	0000004A	D
UCB$L_FQBL	00000004	D	UCB$W_SRCADDR	0000001A	D	VCB$W_MCOUNT	0000004C	D
UCB$L_FQFL	00000000	D	UCB$W_STS	00000058	D	VCB$W_MODE	0000002C	D
UCB$L_FR3	00000010	D	UCB$W_TT_DESIZE	000000A5	D	VCB$W_PENDERR	00000062	D
UCB$L_FR4	00000014	D	UCB$W_UNIT	00000048	D	VCB$W_QUOSIZE	00000060	D
UCB$L_IOQBL	00000044	D	UCB$W_VPROT	0000001A	D	VCB$W_RECORDSZ	00000050	D
UCB$L_IOQFL	00000040	D	VCB$B_BLOCKFACT	00000052	D	VCB$W_RVN	0000000E	D
UCB$L_IRP	0000004C	D	VCB$B_CUR_RVN	0000002F	D	VCB$W_SIZE	00000008	D
UCB$L_LINK	0000002C	D	VCB$B_EOFDELTA	0000004E	D	VCB$W_START_NUM	00000028	D
UCB$L_LOGADR	00000064	D	VCB$B_IBMAPSIZE	00000038	D	VCB$W_START_SEQ	0000002A	D
UCB$L_MAXBLOCK	00000084	D	VCB$B_IBMAPVBN	0000003A	D	VCB$W_TRANS	0000000C	D
UCB$L_MB_MBX	0000007C	D	VCB$B_LRU_LIM	00000049	D	WCB$B_ACCESS	0000000B	D
UCB$L_MB_PORT	0000008C	D	VCB$B_QNAMECNT	0000000B	D	WCB$B_TYPE	0000000A	D
UCB$L_MB_RAST	00000078	D	VCB$B_RESFILES	0000004F	D	WCB$C_LENGTH	00000024	D
UCB$L_MB_SHB	00000080	D	VCB$B_SBMAPSIZE	00000039	D	WCB$C_MAP	00000024	D
UCB$L_MB_WAST	00000074	D	VCB$B_SBMAPVBN	0000003B	D	WCB$K_LENGTH	00000024	D
UCB$L_MB_WIOQBL	00000088	D	VCB$B_STATUS	0000000B	D	WCB$K_MAP	00000024	D
UCB$L_MB_WIOQFL	00000084	D	VCB$B_STATUS2	00000053	D	WCB$L_FCB	00000018	D
UCB$L_MEDIA	0000008C	D	VCB$B_TM	0000002E	D	WCB$L_LBN	00000002	D
UCB$L_NT_DATSSB	00000074	D	VCB$B_TYPE	0000000A	D	WCB$L_ORGUCB	00000010	D
UCB$L_NT_INTSSB	00000078	D	VCB$B_WINDOW	00000048	D	WCB$L_P1_LBN	00000026	D
UCB$L_OPCNT	00000060	D	VCB$C_COMLEN	00000024	D	WCB$L_P2_LBN	0000002C	D
UCB$L_OWNUIC	0000001C	D	VCB$C_LENGTH	00000064	D	WCB$L_PID	0000000C	D
UCB$L_PID	00000028	D	VCB$C_MRKLEN	0000000B	D	WCB$L_PREVLBN	FFFFFFFC	D
UCB$L_RQBL	00000004	D	VCB$K_COMLEN	00000024	D	WCB$L_RVT	0000001C	D
UCB$L_RQFL	00000000	D	VCB$K_LENGTH	00000064	D	WCB$L_STVBN	00000020	D
UCB$L_SVAPTE	00000068	D	VCB$K_MRKLEN	0000000B	D	WCB$L_WLBL	00000004	D
UCB$L_SVPN	00000064	D	VCB$L_AQB	00000010	D	WCB$L_WLFL	00000000	D
UCB$L_TT_DECHAR	000000A8	D	VCB$L_BLOCKBL	00000004	D	WCB$W_ACON	00000014	D
UCB$L_TT_RDUE	0000008C	D	VCB$L_BLOCKFL	00000000	D	WCB$W_COUNT	00000000	D
UCB$L_TT_RTIMOU	00000088	D	VCB$L_CACHE	00000058	D	WCB$W_NMAP	00000016	D
UCB$L_VCB	00000030	D	VCB$L_CUR_FID	00000024	D	WCB$W_P1_COUNT	00000024	D
UCB$T_PARTNER	0000000C	D	VCB$L_FCBBL	00000004	D	WCB$W_P2_COUNT	0000002A	D
UCB$W_BCNT	0000006E	D	VCB$L_FCBFL	00000000	D	WCB$W_PREVCOUNT	FFFFFFFA	D
UCB$W_BCR	00000096	D	VCB$L_FREE	00000040	D	WCB$W_REFCNT	0000000E	D
UCB$W_BOFF	0000006C	D	VCB$L_HOME2LBN	00000028	D	WCB$W_SIZE	00000008	D
UCB$W_BUFQUO	00000018	D	VCB$L_HOMELBN	00000024	D			
UCB$W_BYTESTOGO	0000003E	D	VCB$L_IBMAPLBN	00000030	D			
UCB$W_CHARGE	0000004A	D	VCB$L_IXHDR2LBN	0000002C	D			
UCB$W_CYLINDERS	0000003E	D	VCB$L_MAXFILES	00000044	D			
UCB$W_DA	0000008C	D	VCB$L_MVL	00000034	D			
UCB$W_DC	0000008E	D	VCB$L_QUOCACHE	0000005C	D			
UCB$W_DEVBUFSIZ	0000003A	D	VCB$L_QUOTAFCB	00000054	D			

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000109 (265.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	62.2 (1)	
$MODULE	00000000-R	62.2 (1)	
CCB$L_UCB	00000000		#-103 (1)
CCB$L_WIND	00000004		#-141 (1) #-172 (1)
CCB$W_IOC	0000000A		#-106 (1) #-140 (1) #-165 (1)
CHAN	=00000004	62 (1)	#-98 (1)
DDB$L_DDT	0000000C		#-133 (1)
DDT$L_CANCEL	0000000C		#-135.1 (1)
DEV$V_FOR	=00000018		#-144 (1)
DEV$V_MNT	=00000013		#-143 (1)
DS$AX_SOFTPCB	00000000-XR		97.1 (1)
DYN$C_IRP	=0000000A		#-169 (1)
EXE$ALONONPAGED	00000000-XR		144.2 (1)
EXE$CANCEL	00000007-R	97 (1)	
EXE$QIOACPPKT	00000000-XR		#-192 (1)
IO$_ACPCONTROL	=00000038		#-174 (1)
IOC$GL_PSBL	00000000-XR		123.1 (1)
IOC$VERIFYCHAN	00000000-XR		99 (1)
IPL$_ASTDEL	=00000002		#-105 (1)
IPL$_IOPOST	=00000004		#-126 (1)
IRP$C_LENGTH	0000005C		#-144.1 (1) #-168 (1)
IRP$L_IOQBL	00000004		#-119 (1)
IRP$L_IOQFL	00000000		#-111 (1) 120 (1) 123.1 (1)
IRP$L_MEDIA	00000034		#-123 (1)
IRP$L_PID	0000000C		#-115 (1)
IRP$M_FUNC	=00000002		#-122 (1)
IRP$V_BUFIO	=00000000		#-121 (1)
IRP$V_VIRTUAL	=00000004		#-114 (1)
IRP$W_CHAN	00000028		#-117 (1)
IRP$W_SIZE	00000008		#-167 (1)
IRP$W_STS	0000002A		114 (1) 121 (1) #-122 (1)
PCB$L_PID	00000060		#-115 (1) #-170 (1)
PCB$W_BIOCNT	0000003A		#-164 (1)
PR$_IPL	=00000012		#-105 (1) #-110 (1) #-129 (1) #-131 (1)
PR$_SIRR	=00000014		#-126 (1)
RSN$_NPDYNMEM	=00000003		#-157 (1)
SS$_CANCEL	00000000-XR		#-123 (1)
SS$_INSFMEM	00000000-XR		#-156 (1)
SS$_NORMAL	00000000-XR		#-128 (1)
UCB$B_FIPL	0000000B		#-110 (1) #-131 (1)
UCB$L_DDB	00000024		#-132 (1)
UCB$L_DEVCHAR	00000034		143 (1) 144 (1)
UCB$L_IOQFL	00000040		108 (1)
UCB$L_IRP	0000004C		#-134 (1)
UCB$L_VCB	00000030		#-182 (1)
VCB$W_TRANS	0000000C		#-183 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CADEF	1	40 (1)	40 (1)
$CCBDEF	1	41 (1)	41 (1)
$DDBDEF	1	42 (1)	42 (1)
$DDTDEF	1	43 (1)	43 (1)
$DEFINI	1	40 (1)	40 (1) 41 (1) 42 (1) 43 (1) 44 (1)
			45 (1) 46 (1) 47 (1) 48 (1) 49 (1)
			50 (1) 51 (1) 52 (1) 53 (1) 54 (1)
$DEVDEF	1	44 (1)	44 (1)
$DYNDEF	2	45 (1)	45 (1)
$IODEF	17	46 (1)	46 (1)
$IPLDEF	1	47 (1)	47 (1)
$IRPDEF	4	48 (1)	48 (1)
$PCBDEF	4	49 (1)	49 (1)
$PRDEF	4	50 (1)	50 (1)
$RSNDEF	1	51 (1)	51 (1)
$UCBDEF	10	52 (1)	52 (1)
$VCBDEF	6	53 (1)	53 (1)
$WCBDEF	3	54 (1)	54 (1)
MODNAM	1	62.2 (1)	62.2 (1)
SETIPL	1	105 (1)	105 (1) 110 (1) 129 (1) 131 (1)
SOFTINT	1	126 (1)	126 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.11	00:00:00.26
Command processing	141	00:00:00.78	00:00:02.17
Pass 1	639	00:00:17.36	00:00:33.95
Symbol table sort	0	00:00:01.54	00:00:01.88
Pass 2	98	00:00:02.83	00:00:05.69
Symbol table output	30	00:00:00.20	00:00:00.31
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	15	00:00:00.24	00:00:00.24
Assembler run totals	966	00:00:23.11	00:00:44.55

The working set limit was 1000 pages.
79415 bytes (156 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1076 non-local and 9 local symbols.
188 source lines were read in Pass 1, producing 0 object records in Pass 2.
112 pages of virtual memory were used to define 27 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	11
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	23

1520 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CANCEL/UPDA=(CANCEL.UPD,CANCEL.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(2) 74 DECLARATIONS
(6) 226 $CHANNEL CALL
(7) 286 $CHANNEL CALL - ADAPTER INIT FUNCTION
(8) 319 $CHANNEL CALL - BUS INIT FUNCTION
(9) 352 $CHANNEL CALL - ABORT FUNCTION
(10) 384 $CHANNEL CALL - ENABLE INTERRUPTS
(11) 431 $CHANNEL CALL - PURGE
(12) 463 $CHANNEL CALL - DISABLE INTERRUPTS
(13) 499 $CHANNEL CALL - CLEAR ADAPTER
(14) 531 $CHANNEL CALL - GET ADAPTER STATUS
(15) 573 $CHANNEL CALL - SET DEFEAT PARITY
(16) 615 $CHANNEL CALL - CLEAR DEFEAT PARITY
(17) 657 $CHANNEL CALL - COMMON CALL RETURN
(18) 701 $SETMAP CALL
(19) 1006 DSP$SHOWMBA
(20) 1047 DSP$SHOWUBI
(21) 1091 $SHOWCHAN CALL
(22) 1143 SHOW UBI STATUS
(23) 1172 SHOWBITS
(24) 1210 DETERMINE ADAPTER SPECIFIC STATUS
(25) 1240 DETERMINE GENERAL ADAPTER STATUS
(26) 1270 DETERMINE ADAPTER HARDWARE ERROR STATUS
(28) 1301 BUILD THE CHANNEL DATA BLOCK
(29) 1362 SET ADAPTER VECTOR(S)
(30) 1409 CLEAR ADAPTER VECTOR(S)
(31) 1450 INTERRUPT PRE-PROCESS

```
0000      1          .TITLE  CHAN730 *** CHAN730 Channel services for NEBULA
0000      2          .IDENT  /06-11/
0000      3          .NLIST  CND
0000      4          .LIST   MEB
0000      5
0000      6 ;
0000      7 ; Copyright (c) 1979, 1981, 1983
0000      8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      9 ;
0000     10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     15 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     16 ; REMAIN IN DEC.
0000     17 ;
0000     18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     20 ; CORPORATION.
0000     21 ;
0000     22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     24
0000     25 ;++
0000     26 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
0000     27 ;
0000     28 ; ABSTRACT:
0000     29 ;
0000     30 ; ENVIRONMENT:
0000     31 ;
0000     32 ; AUTHOR:       Roger Riggs     10-Sept-1979    VERSION 01
0000     33 ;
0000     34 ; MODIFIED BY:
0000     35 ;                Roger Riggs, 14-Jan-1979, Version 5.2
0000     36 ;       01       Added lists for CVT$T_.... strings referenced in CVTREG
0000     37 ;       02       Added MTPR #1,#^X37 to do I/O subsystem clear on INITA
0000     38 ;                Roger Riggs, 21-MAY-1980,       Version 5.4
0000     39 ;       03       CLRMAP was clearing 512 map register, only 496 exist
0000     40 ;                Also fixed algorithm for determining how many maps to set
0000     41 ;                from starting address and byte count
0000     42 ;                Roger Riggs, 26-Jun-1980,       Version 5.5
0000     43 ;       04       Fixed bug which did not allocate enough map registers
0000     44 ;                for transfer of 1+n*pages.
0000     45 ;
0000     46 ;       05       - Jack Stansbury, 23-Oct-1981, Version 6.-
0000     47 ;                Removed all calls to $DS_BNERROR and $DS_BERROR since these
0000     48 ;                should be used only by the diagnostics. Replaced them with
0000     49 ;                their BLBx equivalents. Also added .LIBRARY statements for
0000     50 ;                $DS and $DIAG.
0000     51 ;
0000     52 ;       06       - Jack Stansbury, 26-Oct-1981, Version 6.-
0000     53 ;                Changed the .PSECT SEP to WORK, CODE, and DATA
0000     54 ;                where appropriate.
0000     55 ;
0000     56 ;       07       - Dave Butenhof, 23-Nov-1981, verison 6.5
0000     57 ;                Add .LIBRARY statement for DS.MLB, and fixed some
```

```
                        0000     58 ;                truncation errors.
                        0000     59 ;
                        0000     60 ;        08      - Dave Butenhof, 09-May-1981, version 6.5
                        0000     61 ;                Add CHM$_INVALIDATE function to DSX$SETMAP routine
                        0000     62 ;
                        0000     63 ;        09      - Dave Butenhof, 09-May-1981, version 6.5
                        0000     64 ;                Fix assorted register errors
                        0000     65 ;
                        0000     66 ;        10      - Dave Butenhof, 17-Dec-1981, Version 6.6
                        0000     67 ;                Fix problem with map invalidate: it assumed 496 map
                        0000     68 ;                vectors, while there are really 512.
                        0000     69 ;
                        0000     70 ;        11      Bob Bergazzi    May 16,1983     Version 6.11
                        0000     71 ;                Changed the order of the .LIB statements.
                        0000     72 ;--
```

```
                     0000     74          .SBTTL  DECLARATIONS
                     0000     75 ;
                     0000     76 ; INCLUDE FILES:
                     0000     77 ;
                     0000     78
                     0000     79 .LIBRARY         /SYS$LIBRARY:LIB/       ;                                   [11]
                     0000     80 .LIBRARY         /$DS/                   ;                                   [11]
                     0000     81 .LIBRARY         /$DIAG/                 ;                                   [11]
                     0000     82
                     0000     83 ;
                     0000     84 ; MACROS:
                     0000     85 ;
                     0000     86
                     0000     87 .MACRO   $CHNDEF
                     0000     88          $OFFDEF CHN,<UNIT,FUNC,ADR1,ADR2>
                     0000     89 .ENDM    $CHNDEF
                     0000     90 .MACRO   $STMDEF
                     0000     91          $OFFDEF STM,<UNIT,FUNC,PHY,BAS,CNT,PTH>
                     0000     92 .ENDM    $STMDEF
                     0000     93 .MACRO   $SHWDEF
                     0000     94          $OFFDEF SHW,<UNIT>
                     0000     95 .ENDM    $SHWDEF
                     0000     96
                     0000     97 ;
                     0000     98 ; EQUATED SYMBOLS:
                     0000     99 ;
                     0000    100
                     0000    104 $DS_DSDEF
                     0000    105 $UBADEF                                 ; UBA REGISTER DEFINITIONS
                     0000    106 ;$UBIDEF                                    ; UBI REGISTER DEFINITIONS
                     0000    107 ; TEMPORARY UBI DEFINITIONS
00000000             0000    108 UBI$L_CSR       =       ^X0
00000800             0000    109 UBI$L_MAP       =       ^X800
0000001F             0000    110 UBI$V_MAP_VALID =       31
80000000             0000    111 UBI$M_MAP_VALID = 1 @ UBI$V_MAP_VALID
00000019             0000    112 UBI$V_MAP_BO    =       25
02000000             0000    113 UBI$M_MAP_BO    = 1 @ UBI$V_MAP_BO
00000001             0000    114 UBI$S_MAP_BO    =       1
00000015             0000    115 UBI$V_MAP_DPD   =       21
00000002             0000    116 UBI$S_MAP_DPD   =       2
00000000             0000    117 UBI$V_MAP_PFN   =       0
0000000F             0000    118 UBI$S_MAP_PFN   =       15
                     0000    119
                     0000    120
00660081             0000    121 DS$_NOSUPPORT   =       DS$_NOTIMP!1    ; ALLOW UNSUPPORTED
                     0000    122                                         ; FUNCTIONS TO SUCCEDE FOR NOW
                     0000    123 ; END OF TEMPORARY DEFINITIONS
                     0000    124 $DS_BITDEF                              ; BIT DEFINITIONS
                     0000    125 $DS_DSDEF                               ; PROCEDURE RETURN CODE DEFINITIONS
                     0000    126 $DS_PARDEF                              ; PARAMETER ARG DEFINITIONS
                     0000    127 $DS_HPODEF                              ; HARDWARE P-TABLE DEFINITIONS
                     0000    128 $DS_CHDEF                               ; CHANNEL SERVICE DEFINITIONS
                     0000    129 $CHNDEF                                 ; $DS_CHANNEL CALL
                     0000    130 $PRDEF                                  ; PROCESSOR REGISTERS
                     0000    131 $STMDEF                                 ; $DS_SETMAP CALL
                     0000    132 $SHWDEF                                 ; $DS_SHOWCHAN CALL
                     0000    136
```

```
00000080  0000   137 SHW$K_BUFSIZ    =       128             ; $SHOWCHAN BUFFER SIZE
          0000   138
          0000   139 ; INTERNAL FLAG WORD DEFINITIONS
          0000   140
00000000  0000   141 CDB$V_MBA     = 0                       ; ADAPTER IS MBA (11/750,11/780)
00000001  0000   142 CDB$V_UBA     = 1                       ; ADAPTER IS UBA (11/780)
00000002  0000   143 CDB$V_UBI     = 2                       ; ADAPTER IS UBI (11/750)
00000003  0000   144 CDB$V_OFFSET  = 3                       ; OFFSET MODE
00000004  0000   145 CDB$V_REV     = 4                       ; REVERSE OPERATION
00000005  0000   146 CDB$V_IE      = 5                       ; INTERRUPTS ENABLED
00000001  0000   147 CDB$M_MBA     = 1 @ CDB$V_MBA
00000002  0000   148 CDB$M_UBA     = 1 @ CDB$V_UBA
00000004  0000   149 CDB$M_UBI     = 1 @ CDB$V_UBI
00000008  0000   150 CDB$M_OFFSET  = 1 @ CDB$V_OFFSET
00000010  0000   151 CDB$M_REV     = 1 @ CDB$V_REV
00000020  0000   152 CDB$M_IE      = 1 @ CDB$V_IE
          0000   153
          0000   154 ; STATUS BIT DEFINITIONS FOR CDB$L_ST1
          0000   155
0000000F  0000   156 CHS$M_ERRANY     =       CHS$M_SYSERR!CHS$M_CHNERR!CHS$M_DEVERR!CHS$M_PGMERR
          0000   157
```

```
                        00000000   159          .PSECT  Work, NoExe, NoShr, Wrt, Long          ;                    [06]
                            0000   160
                            0000   161 ;
                            0000   162 ; OWN STORAGE:
                            0000   163 ;
                            0000   164
                            0000   165 .SAVE
                            0000   166 .PSECT  $ABS$,ABS
                        00000000   0032   167 .=0
                        00000004   0000   168 CDB$L_FLAGS:     .BLKL   1               ; INTERNAL FLAGS
                        00000008   0004   169 CDB$A_TBL:       .BLKL   1               ; HARDWARE P-TABLE ADDRESS
                        0000000C   0008   170 CDB$A_VEC:       .BLKL   1               ; INTERRUPT VECTOR ADDRESS
                        00000010   000C   171 CDB$A_STORE:     .BLKL   1               ; CALLER STATUS ADDRESS
                        00000014   0010   172 CDB$A_ADDR:      .BLKL   1               ; CHANNEL VECTOR ADDRESS
                        00000018   0014   173 CDB$L_ST1:       .BLKL   1               ; STATUS (HARDWARE ERROR)
                        0000001A   0018   174 CDB$W_ST2:       .BLKW   1               ; STATUS (GENERAL)
                        0000001C   001A   175 CDB$W_RVR:       .BLKW   1               ; RECEIVE VECTOR
                        00000020   001C   176 CDB$L_PTH:       .BLKL   1               ; DATA PATH NUMBER
                        00000020   0020   177 CDBSIZ=.
                                00000000   178 .RESTORE
                            0000   179 CDB$BLOCK::                                     ; CHANNEL CONTROL BLOCK
                        00000020   0000   180 .BLKB   CDBSIZ
                            0020   181
                            0020   182 ;       ARGLISTS FOR $CVTREG AND $PRINTB
                            0020   183 FMT_CVT_ARG:    $DS_CVTREG_L    31,0,0,SHWBUF,SHW$K_BUFSIZ
00000000 00000000 0000001F 0000000B  0020                       .LONG   11, 31, 0, 0
00000000 00000000 00000080 00000068' 0030                       .LONG    SHWBUF, SHW$K_BUFSIZ, 0, 0
00000000 00000000 00000000 00000000  0040                       .LONG    0, 0, 0, 0
                            0050   184
                            0050   185 CVT$T_MBASR::
                        0000003D'   0050   186          .ADDRESS        MBA_SR_BIT      ; Address of Bit mnemonics
                            0054   187                                                  ; No additional arguments needed
                            0054   188 CVT$T_MBAMAP::
                        0000003D'   0054   189          .ADDRESS        MBA_MAP_BIT     ; Address of Bit mnemonics
                            0058   190                                                  ; No additional arguments needed
                            0058   191 CVT$T_UBADPR::
                        0000003D'   0058   192          .ADDRESS        UBA_DPR_BIT     ; Address of Bit mnemonics
                            005C   193                                                  ; No additional arguments needed
                            005C   194 CVT$T_UBAMAP::
                        0000003E'   005C   195          .ADDRESS        UBA_MAP_BIT     ; Address of Bit mnemonics
                            0060   196                                                  ; No additional arguments needed
                            0060   197
                            0060   198 ; COMMON ERROR MESSAGE FOR DEBUG
                            0060   199
                            0060   200 MODNAM  CHANNEL                                 ; DEFINE MODULE NAME FOR ERRSUP
                4C 45 4E 4E 41 48 43 00'  0060      $MODULE:        .ASCIC  \CHANNEL\
                              07  0060
```

```
                                    00000000   202          .PSECT  Data, NoExe, Shr, NoWrt, Long          ;                                [06]
                                        0000   203
                                        0000   204 ; STORAGE FOR $SHOWCHAN - CHANNEL STATUS DUMP
                                        0000   205
                                  2F 21 00'     0000   206 FMT_CRLF:        .ASCIC  .!/.
                                  02            0000
4E 4E 41 48 43 20 49 42 55 2F 21 00'  0003   207 FMT_UBI_HDR:    .ASCIC  .!/UBI CHANNEL STATUS DUMP!/.
55 44 20 53 55 54 41 54 53 20 4C 45   000F
                        2F 21 50 4D   001B
                                 1B   0003
5B 3A 52 53 43 5F 49 42 55 2F 21 00'  001F   208 FMT_UBI_CSR:    .ASCIC  .!/UBI_CSR:[!XL]!_!XL(X);!_!AC.
29 58 28 4C 58 21 5F 21 5D 4C 58 21   002B
                  43 41 21 5F 21 3B   0037
                                 1D   001F
                                      C03D   209 MBA_SR_BIT:
                                      003D   210 MBA_MAP_BIT:
                                      003D   211 UBA_DPR_BIT:
                                  00' 003D   212          .ASCIC  '"'
                                  00  003D
                                      003E   213 UBA_MAP_BIT:
2C 2C 2C 2C 2C 2C 44 49 4C 41 56 00'  003E   214          .ASCIC  "VALID,,,,,,BO,,,,,PFN=21^X"
32 3D 4E 46 50 2C 2C 2C 2C 2C 4F 42   004A
                        58 5E 31      0056
                                 1A   003E
```

```
                    00000068   216          .PSECT  Work, NoExe, NoShr, Wrt, Long          ;                    [06]
                        0068   217
                        0068   218 ;                                                                               [06]
                        0068   219 ; More OWN Storage                                                             [06]
                        0068   220 ;                                                                               [06]
                        0068   221
                000000E9  0068   222 SHWBUF: .BLKB   SHW$K_BUFSIZ+1          ; $SHOWCHAN BUFFER
                000000ED  00E9   223 JMPADR: .BLKL   1                        ; VECTOR FOR ADAPTER/DEVICE INTERRUPTS
                000000F1  00ED   224 UBIVEC: .BLKL   1                        ; TEMP FOR UBI VECTOR
```

```
                    00F1   226          .SBTTL  $CHANNEL CALL
                00000000   227          .PSECT  Code, Exe, Shr, NoWrt, Long                ;                    [06]
                    0000   228 ;++
                    0000   229 ; FUNCTIONAL DESCRIPTION:
                    0000   230 ;
                    0000   231 ;       CHANNEL ADAPTER INTERFACE SERVICE
                    0000   232 ;
                    0000   233 ; CALLING SEQUENCE:
                    0000   234 ;
                    0000   235 ;       CALLG   ARGLST, @#DS$CHANNEL
                    0000   236 ;         OR
                    0000   237 ;       CALLS   #4,     @#DS$CHANNEL
                    0000   238 ;
                    0000   239 ; INPUT PARAMETERS:
                    0000   240 ;
                    0000   241 ;       4(AP)   LOGICAL UNIT NUMBER
                    0000   242 ;       8(AP)   FUNCTION CODE
                    0000   243 ;       12(AP)  VECTOR ADDRESS
                    0000   244 ;       16(AP)  STATUS STORE ADDRESS(ADDRESS OF QWORD)
                    0000   245 ;
                    0000   246 ; IMPLICIT INPUTS:      NONE
                    0000   247 ;
                    0000   248 ; OUTPUT PARAMETERS:    NONE
                    0000   249 ;
                    0000   250 ; IMPLICIT OUTPUTS:     NONE
                    0000   251 ;
                    0000   252 ; COMPLETION CODES:
                    0000   253 ;
                    0000   254 ;       DS$_....
                    0000   255 ;
                    0000   256 ; SIDE EFFECTS:
                    0000   257 ;
                    0000   258 ;       The channel may be affected
                    0000   259 ;--
```

```
                 007C  0000   261 .ENTRY  DSX$CHANNEL,^M<R2,R3,R4,R5,R6>
                       0002   262
      52  04 AC   D0   0002   263         MOVL    CHN$_UNIT(AP),R2                ; GET THE UNIT NUMBER
        0301  30   0006   264         BSBW    BLDCDB                          ; FIND/CREATE THE CDB
      03 50  E8   0009   265         BLBS    R0, CHN_DSP             ; IF ERROR,  RETURN WITH                [05]
                 000C   266                                         ; ERROR CODE FROM BLDCDB                [05]
       00A1  31   000C   267         BRW     CHN_CALL_RTN                    ; RETURN
                 000F   268
                 000F   269 CHN_DSP:
      52  08 AC   D0   000F   270         MOVL    CHN$_FUNC(AP),R2                ; FUNCTION CODE TO R2
                 0013   271         CASE    R2,LIMIT=#0,< -                 ; DISPATCH ON FUNCTION
                 0013   272                 CHN_INITA, -                    ; INITA
                 0013   273                 CHN_INITB, -                    ; INITB
                 0013   274                 CHN_ABORT, -                    ; ABORT
                 C013   275                 CHN_PURGE, -                    ; PURGE
                 0013   276                 CHN_ENINT, -                    ; ENINT
                 0013   277                 CHN_DSINT, -                    ; DSINT
                 0013   278                 CHN_CLEAR, -                    ; CLEAR
                 0013   279                 CHN_STATUS, -                   ; STATUS
                 0013   280                 CHN_SETDFT, -                   ; SET DEFEAT PARITY
                 0013   281                 CHN_CLRDFT, -                   ; CLEAR DEFEAT PARITY
                 0013   282                 >
09' 00  52  AF   0013         CASEW   R2,#0,S^#<<30001$-30000$>/2>-1
                 0017   30000$:
                001E' 0017         .SIGNED_WORD    CHN_INITA-30000$
                002A' 0019         .SIGNED_WORD    CHN_INITB-30000$
                0037' 001B         .SIGNED_WORD    CHN_ABORT-30000$
                005F' 001D         .SIGNED_WORD    CHN_PURGE-30000$
                0040' 001F         .SIGNED_WORD    CHN_ENINT-30000$
                0068' 0021         .SIGNED_WORD    CHN_DSINT-30000$
                006D' 0023         .SIGNED_WORD    CHN_CLEAR-30000$
                0076' 0025         .SIGNED_WORD    CHN_STATUS-30000$
                0080' 0027         .SIGNED_WORD    CHN_SETDFT-30000$
                0089' 0029         .SIGNED_WORD    CHN_CLRDFT-30000$
                       002B   30001$:
50  00660020 8F   D0   002B   283         MOVL    #DS$_PROGERR,R0                 ; INVALID FUNCTION
          007B  31   0032   284         BRW     CHN_CALL_RTN                    ; RETURN
```

```
                                0035   286          .SBTTL  $CHANNEL CALL - ADAPTER INIT FUNCTION
                                0035   287 ;++
                                0035   288 ; FUNCTIONAL DESCRIPTION:
                                C035   289 ;
                                0035   290 ; INITIALIZE ADAPTER HARDWARE
                                0035   291 ;
                                0035   292 ; CALLING SEQUENCE:
                                0035   293 ;
                                0035   294 ; BRW    CHN_INITA
                                0035   295 ;
                                0035   296 ; INPUT PARAMETERS:
                                0035   297 ;
                                0035   298 ;       R5      =       CHANNEL DATA BLOCK
                                0035   299 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                C035   300 ;
                                0035   301 ; IMPLICIT INPUTS:      NONE
                                0035   302 ;
                                0035   303 ; OUTPUT PARAMETERS:    NONE
                                0035   304 ;
                                0035   305 ; IMPLICIT OUTPUTS:     NONE
                                0035   306 ;
                                0035   307 ; COMPLETION CODES:
                                0035   308 ;
                                0035   309 ;       DS$_NORMAL      =       SUCCESS
                                0035   310 ;
                                0035   311 ; SIDE EFFECTS:         NONE
                                0035   312 ;--
                                0035   313
                                0035   314 CHN_INITA:
                     37   01   DA  0035   315          MTPR    #1,#^X37                ; I/O subsystem clear
50   006600B1         8F   D0  0038   316          MOVL    #DS$_NOSUPPORT,R0       ; NOT SUPPORTED
                         68   11  003F   317          BRB     CHN_NORMAL_RTN          ; RETURN
```

```
                                0041   319          .SBTTL  $CHANNEL CALL - BUS INIT FUNCTION
                                0041   320 ;++
                                0041   321 ; FUNCTIONAL DESCRIPTION:
                                0041   322 ;
                                0041   323 ;       NOP FOR 11/730 UNIBUS
                                0041   324 ;
                                0041   325 ; CALLING SEQUENCE:
                                0041   326 ;
                                0041   327 ;       BRW     CHN_INITB
                                0041   328 ;
                                0041   329 ; INPUT PARAMETERS:
                                0041   330 ;
                                0041   331 ;       R5      =       CHANNEL DATA BLOCK
                                0041   332 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                0041   333 ;
                                0041   334 ; IMPLICIT INPUTS:      NONE
                                0041   335 ;
                                0041   336 ; OUTPUT PARAMETERS:    NONE
                                0041   337 ;
                                0041   338 ; IMPLICIT OUTPUTS:     NONE
                                0041   339 ;
                                0041   340 ; COMPLETION CODES:
                                0041   341 ;
                                0041   342 ;       DS$_NOSUPPORT   =       NOT SUPPORTED FOR 11/750 UBI OR 11/730
                                0041   343 ;
                                0041   344 ; SIDE EFFECTS: NONE
                                0041   345 ;--
                                0041   346
                                0041   347 CHN_INITB:
          37   01    DA         0041   348          MTPR    #1,#^X37                ; I/O subsystem clear
50   006600B1 8F    D0         0044   349          MOVL    #DS$_NOSUPPORT,R0       ; FUNCTION NOT SUPPORTED
          0062    31           004B   350          BRW     CHN_CALL_RTN            ; RETURN
```

```
                     004E    352          .SBTTL  $CHANNEL CALL - ABORT FUNCTION
                     004E    353 ;++
                     004E    354 ; FUNCTIONAL DESCRIPTION:
                     004E    355 ;
                     004E    356 ;       NOP FOR 11/730
                     004E    357 ;
                     004E    358 ; CALLING SEQUENCE:
                     004E    359 ;
                     004E    360 ;       BRW     CHN_ABORT
                     004E    361 ;
                     004E    362 ; INPUT PARAMETERS:
                     004E    363 ;
                     004E    364 ;       R5      =       CHANNEL DATA BLOCK
                     004E    365 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     004E    366 ;
                     004E    367 ; IMPLICIT INPUTS:      NONE
                     004E    368 ;
                     004E    369 ; OUTPUT PARAMETERS:    NONE
                     004E    370 ;
                     004E    371 ; IMPLICIT OUTPUTS:     NONE
                     004E    372 ;
                     004E    373 ; COMPLETION CODES:
                     004E    374 ;
                     004E    375 ;       DS$_NOSUPPORT   =       FUNCTION NOT SUPPORTED
                     004E    376 ;
                     004E    377 ; SIDE EFFECTS:         NONE
                     004E    378 ;--
                     004E    379
                     004E    380 CHN_ABORT:
 50  006600B1 8F    D0  004E    381          MOVL    #DS$_NOSUPPORT,R0       ; INDICATE NO SUPPORT
                  59    11  0055    382          BRB     CHN_CALL_RTN            ; RETURN
```

```
                                0057   384          .SBTTL  $CHANNEL CALL - ENABLE INTERRUPTS
                                0057   385 ;++
                                0057   386 ; FUNCTIONAL DESCRIPTION:
                                0057   387 ;
                                0057   388 ;       ENABLE ADAPTER INTERRUPTS
                                0057   389 ;
                                0057   390 ; CALLING SEQUENCE:
                                0057   391 ;
                                0057   392 ;       BRB     CHN_ENINT
                                0057   393 ;
                                0057   394 ; INPUT PARAMETERS:
                                0057   395 ;
                                0057   396 ;       R5      =       CHANNEL DATA BLOCK
                                0057   397 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                0057   398 ;
                                0057   399 ; IMPLICIT INPUTS:      NONE
                                0057   400 ;
                                0057   401 ; OUTPUT PARAMETERS:    NONE
                                0057   402 ;
                                0057   403 ; IMPLICIT OUTPUTS:     NONE
                                0057   404 ;
                                0057   405 ; COMPLETION CODES:
                                0057   406 ;
                                0057   407 ;       DS$_NORMAL      =       SUCCESS
                                0057   408 ;       DS$_IVVECT      =       INVALID VECTOR ($SETVEC)
                                0057   409 ;       DS$_IVADDR      =       INVALID ADDRESS ($SETVEC)
                                0057   410 ;
                                0057   411 ; SIDE EFFECTS:
                                0057   412 ;
                                0057   413 ;       CALLS $SETVEC TO POINT ADAPTER/UUT INTERRUPTS INTO CHANNEL SERVICE
                                0057   414 ;
                                0057   415 ;--
                                0057   416
                                0057   417 CHN_ENINT:
                     02EE   30  0057   418          BSBW    SETVEC                  ; SET THE VECTOR'S
                   17 50    E9  005A   419          BLBC    R0,40$                  ; IF ERROR RETURN WITH IT
                               005D   420
  50   00660020 8F    D0       005D   421          MOVL    #DS$_PROGERR,R0         ; ASSUME ADDRESS CHECK FAILED
                               0064   422
     08 A5    0C AC    D0       0064   423          MOVL    CHN$_ADR1(AP),CDB$A_VEC(R5)     ; SAVE CALLERS VECTOR
                 09    15       0069   424          BLEQ    40$                     ; ZERO OR NEGATIVE IS NOGOOD
     0C A5    10 AC    D0       006B   425          MOVL    CHN$_ADR2(AP),CDB$A_STORE(R5)   ; SAVE CALLERS STORE ADDRESS
                 02    15       0070   426          BLEQ    40$                     ; GTR ZERO IS CORRECT
                 50    D6       0072   427          INCL    R0                      ; Flag success
                               0074   428
                 3A    11       0074   429 40$:     BRB     CHN_CALL_RTN            ; RETURN
```

```
                        0076   431          .SBTTL  $CHANNEL CALL - PURGE
                        0076   432 ;++
                        0076   433 ; FUNCTIONAL DESCRIPTION:
                        0076   434 ;
                        0076   435 ;       NOP FOR 11/730
                        0076   436 ;
                        0076   437 ; CALLING SEQUENCE:
                        0076   438 ;
                        0076   439 ;       BRW     CHN_PURGE
                        0076   440 ;
                        0076   441 ; INPUT PARAMETERS:
                        0076   442 ;
                        0076   443 ;       R5      =       CHANNEL DATA BLOCK
                        0076   444 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                        0076   445 ;
                        0076   446 ; IMPLICIT INPUTS:      NONE
                        0076   447 ;
                        0076   448 ; OUTPUT PARAMETERS:    NONE
                        0076   449 ;
                        0076   450 ; IMPLICIT OUTPUTS:     NONE
                        0076   451 ;
                        0076   452 ; COMPLETION CODES:
                        0076   453 ;
                        0076   454 ;       DS$_NOSUPPORT   = not supported
                        0076   455 ;
                        0076   456 ; SIDE EFFECTS:         NONE
                        0076   457 ;--
                        0076   458
                        0076   459 CHN_PURGE:
      50    006600B1 8F   D0  0076   460          MOVL    #DS$_NOSUPPORT,R0       ; Not supported
                     31   11  007D   461          BRB     CHN_CALL_RTN            ; Return
```

```
                     007F    463          .SBTTL  $CHANNEL CALL - DISABLE INTERRUPTS
                     007F    464 ;++
                     007F    465 ; FUNCTIONAL DESCRIPTION:
                     007F    466 ;
                     007F    467 ;       DISABLE ADAPTER INTERRUPT PROCESSING
                     007F    468 ;
                     007F    469 ; CALLING SEQUENCE:
                     007F    470 ;
                     007F    471 ;       BRW     CHN_DSINT
                     007F    472 ;
                     007F    473 ; INPUT PARAMETERS:
                     007F    474 ;
                     007F    475 ;       R5      =       CHANNEL DATA BLOCK
                     007F    476 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     007F    477 ;
                     007F    478 ; IMPLICIT INPUTS:      NONE
                     007F    479 ;
                     007F    480 ; OUTPUT PARAMETERS:    NONE
                     007F    481 ;
                     007F    482 ; IMPLICIT OUTPUTS:     NONE
                     007F    483 ;
                     007F    484 ; COMPLETION CODES:
                     007F    485 ;
                     007F    486 ;       DS$_NORMAL      =       SUCCESS
                     007F    487 ;       DS$_IVVECT      =       INVALID VECTOR ($CLRVEC)
                     007F    488 ;
                     007F    489 ; SIDE EFFECTS:
                     007F    490 ;
                     007F    491 ;       CALLS $CLRVEC
                     007F    492 ;
                     007F    493 ;--
                     007F    494
                     007F    495 CHN_DSINT:
  02FA  30           007F    496          BSBW    CLRVEC                  ; CLEAR VECTOR'S
    20  11           0082    497          BRB     CHN_CALL_RTN            ; RETURN WITH STATUS IN R0
```

```
                     0084   499          .SBTTL  $CHANNEL CALL - CLEAR ADAPTER
                     0084   500 ;++
                     0084   501 ; FUNCTIONAL DESCRIPTION:
                     0084   502 ;
                     0084   503 ;       CLEAR ADAPTER STATUS
                     0084   504 ;
                     0084   505 ; CALLING SEQUENCE:
                     0084   506 ;
                     0084   507 ;       BRW     CHN_CLEAR
                     0084   508 ;
                     0084   509 ; INPUT PARAMETERS:
                     0084   510 ;
                     0084   511 ;       R5      =       CHANNEL DATA BLOCK
                     0084   512 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0084   513 ;
                     0084   514 ; IMPLICIT INPUTS:      NONE
                     0084   515 ;
                     0084   516 ; OUTPUT PARAMETERS:    NONE
                     0084   517 ;
                     0084   518 ; IMPLICIT OUTPUTS:     NONE
                     0084   519 ;
                     0084   520 ; COMPLETION CODES:
                     0084   521 ;
                     0084   522 ;       DS$_NOSUPPORT   = Function NOPED
                     0084   523 ;
                     0084   524 ; SIDE EFFECTS:         NONE
                     0084   525 ;--
                     0084   526
                     0084   527 CHN_CLEAR:
 50  006600B1 8F     D0   0084   528          MOVL    #DS$_NOSUPPORT,R0       ; Not supported
          23 11      11   008B   529          BRB     CHN_CALL_RTN
```

```
                                008D   531          .SBTTL  $CHANNEL CALL - GET ADAPTER STATUS
                                008D   532 ;++
                                008D   533 ; FUNCTIONAL DESCRIPTION:
                                008D   534 ;
                                008D   535 ;       PASS ADAPTER STATUS TO CALLER
                                008D   536 ;
                                008D   537 ; CALLING SEQUENCE:
                                008D   538 ;
                                008D   539 ;       BRW     CHN_STATUS
                                008D   540 ;
                                008D   541 ; INPUT PARAMETERS:
                                008D   542 ;
                                008D   543 ;       R5      =       CHANNEL DATA BLOCK
                                008D   544 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                C08D   545 ;
                                008D   546 ; IMPLICIT INPUTS:
                                008D   547 ;
                                008D   548 ;       NONE
                                008D   549 ;
                                008D   550 ; OUTPUT PARAMETERS:
                                008D   551 ;
                                008D   552 ;       R0      =       COMPLETION CODE
                                008D   553 ;
                                008D   554 ; IMPLICIT OUTPUTS:
                                008D   555 ;
                                008D   556 ;       NONE
                                008D   557 ;
                                008D   558 ; COMPLETION CODES:
                                008D   559 ;
                                008D   560 ;       DS$_NORMAL      =       SUCCESS
                                008D   561 ;
                                008D   562 ; SIDE EFFECTS:
                                008D   563 ;
                                008D   564 ;       NONE
                                008D   565 ;
                                008D   566 ;--
                                008D   567
                                008D   568 CHN_STATUS:
                         14 A5   7C  008D   569          CLRQ    CDB$L_ST1(R5)                   ; Clear all status
                   10 BC 14 A5   7D  0090   570          MOVQ    CDB$L_ST1(R5),@CHN$_ADR2(AP)    ; PASS STATUS
                      12         11  0095   571          BRB     CHN_NORMAL_RTN                  ; RETURN
```

```
                     0097   573          .SBTTL  $CHANNEL CALL - SET DEFEAT PARITY
                     0097   574 ;++
                     0097   575 ; FUNCTIONAL DESCRIPTION:
                     0097   576 ;
                     0097   577 ;       SET DEFEAT DATA PATH PARITY
                     0097   578 ;
                     0097   579 ; CALLING SEQUENCE:
                     0097   580 ;
                     0097   581 ;       BSBW    CHN_SETDFT
                     0097   582 ;
                     0097   583 ; INPUT PARAMETERS:
                     0097   584 ;
                     0097   585 ;       R5      =       CHANNEL DATA BLOCK
                     0097   586 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0097   587 ;
                     0097   588 ; IMPLICIT INPUTS:
                     0097   589 ;
                     0097   590 ;       NONE
                     0097   591 ;
                     0097   592 ; OUTPUT PARAMETERS:
                     0097   593 ;
                     0097   594 ;       R0      =       COMPLETION CODE
                     0097   595 ;
                     0097   596 ; IMPLICIT OUTPUTS:
                     0097   597 ;
                     0097   598 ;       NONE
                     0097   599 ;
                     0097   600 ; COMPLETION CODES:
                     0097   601 ;
                     0097   602 ;       DS$_NORMAL      =       SUCCESS
                     0097   603 ;       DS$_NOSUPPORT   =       FUNCTION NOT SUPPORTED FOR 11/750 UBI
                     0097   604 ;
                     0097   605 ; SIDE EFFECTS:
                     0097   606 ;
                     0097   607 ;       NONE
                     0097   608 ;
                     0097   609 ;--
                     0097   610
                     0097   611 CHN_SETDFT:
50   00660081 8F     D0  0097   612          MOVL    #DS$_NOSUPPORT,R0       ; INDICATE NO SUPPORT
                 10  11  009E   613          BRB     CHN_CALL_RTN            ; RETURN
```

```
                                00A0   615          .SBTTL  $CHANNEL CALL - CLEAR DEFEAT PARITY
                                00A0   616 ;++
                                00A0   617 ; FUNCTIONAL DESCRIPTION:
                                00A0   618 ;
                                00A0   619 ;       CLEAR DEFEAT DATA PATH PARITY (UBI ONLY)
                                00A0   620 ;
                                00A0   621 ; CALLING SEQUENCE:
                                00A0   622 ;
                                00A0   623 ;       BRW     CHN_CLRDFT
                                00A0   624 ;
                                00A0   625 ; INPUT PARAMETERS:
                                00A0   626 ;
                                00A0   627 ;       R5      =       CHANNEL DATA BLOCK
                                00A0   628 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                00A0   629 ;
                                00A0   630 ; IMPLICIT INPUTS:
                                00A0   631 ;
                                00A0   632 ;       NONE
                                00A0   633 ;
                                00A0   634 ; OUTPUT PARAMETERS:
                                00A0   635 ;
                                00A0   636 ;       R0      =       COMPLETION CODE
                                00A0   637 ;
                                00A0   638 ; IMPLICIT OUTPUTS:
                                00A0   639 ;
                                00A0   640 ;       NONE
                                00A0   641 ;
                                00A0   642 ; COMPLETION CODES:
                                00A0   643 ;
                                00A0   644 ;       DS$_NORMAL      =       SUCCESS
                                00A0   645 ;       DS$_NOSUPPORT   =       NOT SUPPORTED (11/750 UBI)
                                00A0   646 ;
                                00A0   647 ; SIDE EFFECTS:
                                00A0   648 ;
                                00A0   649 ;       NONE
                                00A0   650 ;
                                00A0   651 ;--
                                00A0   652
                                00A0   653 CHN_CLRDFT:
 50   006600B1 8F    D0         00A0   654          MOVL    #DS$_NOSUPPORT,R0               ; NOT SUPPORTED
                  07    11      00A7   655          BRB     CHN_CALL_RTN                    ; RETURN
```

```
                     00A9   657          .SBTTL  $CHANNEL CALL - COMMON CALL RETURN
                     00A9   658 ;++
                     00A9   659 ; FUNCTIONAL DESCRIPTION:
                     00A9   660 ;
                     00A9   661 ;       COMMON CALL RETURN
                     00A9   662 ;
                     00A9   663 ; CALLING SEQUENCE:
                     00A9   664 ;
                     00A9   665 ;       BRW     CHN_CALL_RTN
                     00A9   666 ;
                     00A9   667 ; INPUT PARAMETERS:
                     00A9   668 ;
                     00A9   669 ;       R0      =       COMPLETION CODE
                     00A9   670 ;       R5      =       CHANNEL DATA BLOCK
                     00A9   671 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     00A9   672 ;
                     00A9   673 ; IMPLICIT INPUTS:
                     00A9   674 ;
                     00A9   675 ;       NONE
                     00A9   676 ;
                     00A9   677 ; OUTPUT PARAMETERS:
                     00A9   678 ;
                     00A9   679 ;       R0      =       COMPLETION CODE
                     00A9   680 ;
                     00A9   681 ; IMPLICIT OUTPUTS:
                     00A9   682 ;
                     00A9   683 ;       NONE
                     00A9   684 ;
                     00A9   685 ; COMPLETION CODES:
                     00A9   686 ;
                     00A9   687 ;       NONE
                     00A9   688 ;
                     00A9   689 ; SIDE EFFECTS:
                     00A9   690 ;
                     00A9   691 ;       NONE
                     00A9   692 ;
                     00A9   693 ;--
                     00A9   694
                     00A9   695 CHN_NORMAL_RTN:
 50  00660001 8F  D0 00A9   696          MOVL    #DS$_NORMAL,R0                  ; FLAG NORMAL RETURN
                     0080   697
                     0080   698 CHN_CALL_RTN:
                  04 0080   699          RET                                     ; RETURN
```

```
                        .SBTTL  $SETMAP CALL
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       SET ADPATER MAPPING
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLG   ARGLST, @#DSX$SETMAP
                ;         OR
                ;       CALLS   #6,     @#DSX$SETMAP
                ;
                ; INPUT PARAMETERS:
                ;
                ;       4(AP)   LOGICAL UNIT NUMBER
                ;       8(AP)   FUNCTION CODE
                ;       12(AP)  PHYSICAL ADDRESS(POINTER TO QWORD)
                ;       16(AP)  BASE ADDRESS(TO START MAPPING)
                ;       20(AP)  BYTE COUNT
                ;       24(AP)  DATA PATH NUMBER
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R0      =       COMPLETION CODE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
                ;       DS$_NORMAL      =       SUCCESS
                ;       DS$_PROGERR     =       PROGRAM ERROR
                ;                                MISSING/INVALID P-TABLE
                ;                                ADDRESS/BYTE COUNT INVALID
                ;                                INVALID DATA PATH
                ;       DS$_IHWE        =       INITIAL ADAPTER HARDWARE ERROR ENCOUNTERED
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ;--
```

```
                     0FFC  00B1   749 .ENTRY  DSX$SETMAP,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                           00B3   750
          52  04 AC    D0  00B3   751         MOVL    STM$_UNIT(AP),R2                ; LOGICAL UNIT TO R2
              0250     30  00B7   752         BSBW    BLDCDB                          ; BUILD THE CDB
              03 50    E8  00BA   753         BLBS    R0, 10$                 ; ERROR ?                       [05]
               01C0    31  00BD   754         BRW     STM_CALL_RTN                    ; YES - RETURN
                           00C0   755
          52  0C AC    D0  00C0   756 10$:    MOVL    STM$_PHY(AP),R2                 ; GET THE PHYSICAL ADDRESS
               0184    30  00C4   757         BSBW    RNGCHK                          ; RANGE CHECK
              03 50    E8  00C7   758         BLBS    R0, 20$                 ; RANGE CHECK ERROR ?           [05]
               01B3    31  00CA   759         BRW     STM_CALL_RTN                    ; YES-
                           00CD   760
      1C A5   18 AC    D0  00CD   761 20$:    MOVL    STM$_PTH(AP),CDB$L_PTH(R5)      ; REMEMBER THE DATA PATH
        52    10 AC    D0  00D2   762         MOVL    STM$_BAS(AP),R2                 ; YES- BASE ADDRESS TO R2
        53    0C BC    D0  00D6   763         MOVL    @STM$_PHY(AP),R3                ; START PHYSICAL ADDRESS TO R3
        5A    14 AC    D0  00DA   764         MOVL    STM$_CNT(AP),R10                ; BYTE COUNT TO R10
  50    53    17   09    EF  00DE   765         EXTZV   #9,#23,R3,R0                    ; Get PFN of start page
     58    01FF CA43    9E  00E3   766         MOVAB   511(R10)[R3],R8                 ; Last byte R10+R3-1 round to page
  58    58    17   09    EF  00E9   767         EXTZV   #9,#23,R8,R8                    ; Get PFN of last page
           58    50    C2  00EE   768         SUBL    R0,R8                           ; End-start = maps to alloc
              013B    30  00F1   769         BSBW    BASCHK                          ; CHECK BASE
              03 50   E8  00F4   770         BLBS    R0,60$                          ; IF ERROR RETURN WITH
               0186   31  00F7   771         BRW     STM_CALL_RTN                    ; BASCHK COMPLETION CODE
                           00FA   772 60$:
               0152   30  00FA   773         BSBW    CNTCHK                          ; CHECK COUNT
              03 50   E8  00FD   774         BLBS    R0,70$                          ; IF ERROR RETURN WITH
               017D   31  0100   775         BRW     STM_CALL_RTN                    ; CNTCHK COMPLETION CODE
                          0103   776 70$:
     50   08 AC    D0  0103   777         MOVL    STM$_FUNC(AP),R0                ; GET THE FUNCTION CODE
                          0107   778         CASE    R0,LIMIT=#0,TYPE=L,<-           ;
                          0107   779         NFWDN,-                                 ; NOMAP-FORWARD-NOVALIDATE
                          0107   780         INVALIDATE,-                            ; Invalidate all registers      [08]
                          0107   781         MFWDN,-                                 ; MAP-FORWARD-NOVALIDATE
                          0107   782         MFWDV,-                                 ; MAP-FORWARD-VALIDATE
                          0107   783         NREVN,-                                 ; NOMAP-REVERSE-NOVALIDATE
                          0107   784         INVALID,-                               ; INVALID
                          0107   785         MREVN,-                                 ; MAP-REVERSE-NOVALIDATE
                          0107   786         MREVV,-                                 ; MAP-REVERSE-VALIDATE
                          0107   787         INVALID,-                               ; INVALID
                          0107   788         INVALID,-                               ; INVALID
                          0107   789         MFWDNO,-                                ; MAP-FORWARD-NOVALIDATE-OFFSET
                          0107   790         MFWDVO,-                                ; MAP-FORWARD-VALIDATE-OFFSET
                          0107   791         INVALID,-                               ; INVALID
                          0107   792         INVALID,-                               ; INVALID
                          0107   793         MREVNO,-                                ; MAP-REVERSE-NOVALIDATE-OFFSET
                          0107   794         MREVVO,-                                ; MAP-REVERSE-VALIDATE-OFFSET
                          0107   795         >
OF'  00   50    CF        0107              CASEL   R0,#0,S^#<<30003$-30002$>/2>-1
                          0108        30002$:
                0054'     0108              .SIGNED_WORD    NFWDN-30002$
                002A'     010D              .SIGNED_WORD    INVALIDATE-30002$
                0045'     010F              .SIGNED_WORD    MFWDN-30002$
                0033'     0111              .SIGNED_WORD    MFWDV-30002$
                0087'     0113              .SIGNED_WORD    NREVN-30002$
                0020'     0115              .SIGNED_WORD    INVALID-30002$
                0075'     0117              .SIGNED_WORD    MREVN-30002$
                0060'     0119              .SIGNED_WORD    MREVV-30002$
```

```
                     0020' 011B                .SIGNED_WORD    INVALID-30002$
                     0020' 011D                .SIGNED_WORD    INVALID-30002$
                     00AB' 011F                .SIGNED_WORD    MFWDNO-30002$
                     0096' 0121                .SIGNED_WORD    MFWDVO-30002$
                     0020' 0123                .SIGNED_WORD    INVALID-30002$
                     0020' 0125                .SIGNED_WORD    INVALID-30002$
                     00CF' 0127                .SIGNED_WORD    MREVNO-30002$
                     00BD' 0129                .SIGNED_WORD    MREVVO-30002$
                           012B          30003$:
                           012B    796 INVALID:
          50  00660020 8F   D0  012B    797           MOVL    #DS$_PROGERR,R0                 ; INVALID FUNCTION CODE
                       014B   31  0132    798           BRW     STM_CALL_RTN                    ; RETURN
                           0135    799 INVALIDATE:
                     65    18   CA  0135    800           BICL    #CDB$M_OFFSET ! CDB$M_REV, -    ;                               [08]
                           0138    801                    CDB$L_FLAGS(R5)                 ;                               [08]
                       00E2   30  0138    802           BSBW    CLRMAP                          ; Clear all map regs            [08]
                       013B   31  013B    803           BRW     STM_NORMAL_RTN                  ; Return OK                     [08]
                           013E    804 MFWDV:
                         18   CA  013E    805           BICL    #CDB$M_OFFSET!CDB$M_REV,-
                         65       0140    806                    CDB$L_FLAGS(R5)                 ; CLEAR OFFSET AND REVERSE FLAGS
                       00D9   30  0141    807           BSBW    CLRMAP                          ; CLEAR ALL MAPS
                       00A2   30  0144    808           BSBW    SETMAP                          ; SET MAPS FOR FORWARD TRANSFER
                       00E3   30  0147    809           BSBW    SETVAR                          ; SET VAR IF MBA
                       00E1   30  014A    810           BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                       0129   31  014D    811           BRW     STM_NORMAL_RTN                  ; RETURN
                           0150    812
                           0150    813 MFWDN:
                         18   CA  0150    814           BICL    #CDB$M_OFFSET!CDB$M_REV,-
                         65       0152    815                    CDB$L_FLAGS(R5)                 ; CLEAR OFFSET AND REVERSE FLAGS
                       0093   30  0153    816           BSBW    SETMAP                          ; SET MAPS FOR FORWARD TRANSFER
                       00D4   30  0156    817           BSBW    SETVAR                          ; SET VAR IF MBA
                       00D2   30  0159    818           BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                       011A   31  015C    819           BRW     STM_NORMAL_RTN                  ; RETURN
                           015F    820
                           015F    821 NFWDN:
                         18   CA  015F    822           BICL    #CDB$M_OFFSET!CDB$M_REV,-
                         65       0161    823                    CDB$L_FLAGS(R5)                 ; CLEAR OFFSET AND REVERSE FLAGS
                       00C8   30  0162    824           BSBW    SETVAR                          ; SET VAR IF MBA
                       00C6   30  0165    825           BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                       010E   31  0168    826           BRW     STM_NORMAL_RTN                  ; RETURN
                           016B    827
                           016B    828 MREVV:
                     65    08   CA  016B    829           BICL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)   ; CLEAR OFFSET FLAG
                     65    10   C8  016E    830           BISL    #CDB$M_REV,CDB$L_FLAGS(R5)      ; SET REVERSE FLAG
                       00A9   30  0171    831           BSBW    CLRMAP                          ; CLEAR ALL MAP ENTRIES
                       0072   30  0174    832           BSBW    SETMAP                          ; SET MAPS FOR REVERSE OPERATION
                       00B3   30  0177    833           BSBW    SETVAR                          ; SET VAR IF MBA
                       00B1   30  017A    834           BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                       00F9   31  017D    835           BRW     STM_NORMAL_RTN                  ; RETURN
                           0180    836
                           0180    837 MREVN:
                     65    08   CA  0180    838           BICL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)   ; CLEAR OFFSET FLAG
                     65    10   C8  0183    839           BISL    #CDB$M_REV,CDB$L_FLAGS(R5)      ; SET REVERSE FLAG
                       0060   30  0186    840           BSBW    SETMAP                          ; SET MAPPING
                       00A1   30  0189    841           BSBW    SETVAR                          ; SET VAR IF MBA
                       009F   30  018C    842           BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                       00E7   31  018F    843           BRW     STM_NORMAL_RTN                  ; RETURN
```

```
                        0192   844
                        0192   845 NREVN:
65   08   CA            0192   846          BICL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)    ; CLEAR OFFSET FLAG
65   10   C8            0195   847          BISL    #CDB$M_REV,CDB$L_FLAGS(R5)       ; SET REVERSE FLAG
     0092 30            0198   848          BSBW    SETVAR                           ; SET VAR IF MBA
     0090 30            019B   849          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
     00D8 31            019E   850          BRW     STM_NORMAL_RTN                   ; RETURN
                        01A1   851
                        01A1   852 MFWDVO:
65   10   CA            01A1   853          BICL    #CDB$M_REV,CDB$L_FLAGS(R5)       ; CLEAR REVERSE FLAG
65   08   C8            01A4   854          BISL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)    ; SET OFFSET FLAG
     0073 30            01A7   855          BSBW    CLRMAP                           ; INVALIDATE MAPS
     003C 30            01AA   856          BSBW    SETMAP                           ; SET MAPS
     007D 30            01AD   857          BSBW    SETVAR                           ; SET VAR IF MBA
     007B 30            01B0   858          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
     00C3 31            01B3   859          BRW     STM_NORMAL_RTN                   ; RETURN
                        01B6   860
                        01B6   861 MFWDNO:
65   10   CA            01B6   862          BICL    #CDB$M_REV,CDB$L_FLAGS(R5)       ; CLEAR REVERSE FLAG
65   08   C8            01B9   863          BISL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)    ; SET OFFSET FLAG
     002A 30            01BC   864          BSBW    SETMAP                           ; SET MAPPING
     006B 30            01BF   865          BSBW    SETVAR                           ; SET VAR IF MBA
     0069 30            01C2   866          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
     0081 31            01C5   867          BRW     STM_NORMAL_RTN                   ; RETURN
                        01C8   868
                        01C8   869 MREVVO:
18   C8                 01C8   870          BISL    #CDB$M_OFFSET!CDB$M_REV,-
65                      01CA   871                  CDB$L_FLAGS(R5)                  ; SET OFFSET AND REVERSE FLAGS
     004F 30            01CB   872          BSBW    CLRMAP                           ; CLEAR MAPS
     0018 30            01CE   873          BSBW    SETMAP                           ; SET MAPPING
     0059 30            01D1   874          BSBW    SETVAR                           ; SET VAR IF MBA
     0057 30            01D4   875          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
     009F 31            01D7   876          BRW     STM_NORMAL_RTN                   ; RETURN
                        01DA   877
                        01DA   878 MREVNO:
18   C8                 01DA   879          BISL    #CDB$M_OFFSET!CDB$M_REV,-
65                      01DC   880                  CDB$L_FLAGS(R5)                  ; SET OFFSET AND REVERSE FLAGS
     0009 30            01DD   881          BSBW    SETMAP                           ; SET MAPPING
     004A 30            01E0   882          BSBW    SETVAR                           ; SET VAR IF MBA
     0048 30            01E3   883          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
     0090 31            01E6   884          BRW     STM_NORMAL_RTN                   ; RETURN
                        01E9   885
```

```
                                01E9    887 ;+
                                01E9    888 ; ROUTINE TO SET ADAPTER MAP REGISTERS
                                01E9    889 ; CALLED BY: BSBW SETMAP
                                01E9    890 ; R2 = BASE ADDRESS TO START MAPPING
                                01E9    891 ; R3 = START PHYSICAL ADDRESS
                                01E9    892 ; R6 = ADAPTER CSR ADDRESS
                                01E9    893 ; R8 = NUMBER OF PAGES TO MAP
                                01E9    894 ;-
                                01E9    895 SETMAP:
            51    52    D0      01E9    896         MOVL    R2,R1                           ; BASE TO R1
                  08    BB      01EC    897         PUSHR   #^M<R3>                         ; SAVE R3
     53   53    F7    8F      01EE    898         ASHL    #-9,R3,R3                       ; SHIFT PFN
                  50    D4      01F3    899         CLRL    R0                              ; INIT R0
     50   80000000    8F      C8      01F5    900         BISL    #UBI$M_MAP_VALID,R0             ; SET THE VALID BIT
            65    08    D3      01FC    901         BITL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)   ; OFFSET MODE ?
                  07    13      01FF    902         BEQL    10$                             ; NO -
     50   02000000    8F      C8      0201    903         BISL    #UBI$M_MAP_BO,R0                ; YES- SET THE BIT
                                0208    904 10$:
            00    53    F0      0208    905         INSV    R3,#UBA$V_MAP_ADDR,-            ; INSERT THE PFN
            50    15            0208    906                 #UBA$S_MAP_ADDR,R0              ; INTO R0
      0800 C641    50    D0      020D    907         MOVL    R0,UBI$L_MAP(R6)[R1]            ; CREATE THE MAP
                  53    D6      0213    908         INCL    R3                              ; INC THE ENTRY
                  51    D6      0215    909         INCL    R1                              ; INC THE POINTER
                EE  58    F5      0217    910         SOBGTR  R8,10$                          ; DO ALL PAGES
                  08    BA      021A    911         POPR    #^M<R3>                         ; RESTORE R3
                                021C    912
                        05      021C    913         RSB                                     ; RETURN
                                021D    914
```

B 1 SCH$ASTDEL - AST DELIVERY INTE
C 1 SCH$ASTDEL - AST DELIVERY INTE
D 1 SCH$QAST - ENQUEUE AST CONTROL
E 1 SCH$QAST - ENQUEUE AST CONTROL
F 1 Symbol table
G 1 Psect synopsis
H 1 Cross reference
I 1 Cross reference
J 1 Cross reference
K 1 Cross reference
L 1 *** ATTACH Handle ATTACH comma
M 1 *** ATTACH Handle ATTACH comma
N 1 *** ATTACH Handle ATTACH comma
B 2 *** ATTACH Handle ATTACH comma
C 2 Libraries, Equated Symbols, Ex
D 2 Data Psect Declarations
E 2 Data Psect Declarations
F 2 Data Psect Declarations
G 2 Work Psect Declarations
H 2 DSV$ATTACH Attach command proc
I 2 DSV$ATTACH Attach command proc
J 2 DSV$ATTACH Attach command proc
K 2 DSX$ATTACH Process ATTACH comm
L 2 DSX$ATTACH Process ATTACH comm
M 2 DSX$ATTACH Process ATTACH comm
N 2 DSX$ATTACH Process ATTACH comm
B 3 Process invalid generic name e
C 3 Process invalid generic name e
D 3 Process invalid generic name e
E 3 Get name processing
F 3 Get name processing
G 3 Get name processing
H 3 Get name processing
I 3 Get device attributes
J 3 Get device attributes
K 3 parse device name
L 3 parse device name
M 3 parse device name
N 3 parse device name
B 4 PT_INTERPRET Decode PT-descrip
C 4 PT_INTERPRET Decode PT-descrip
D 4 PT_INTERPRET Decode PT-descrip
E 4 PT_INTERPRET Decode PT-descrip
F 4 PT_INTERPRET Decode PT-descrip
G 4 PT_INTERPRET Decode PT-descrip
H 4 PT_INTERPRET Decode PT-descrip
I 4 DP$EXTRACT Make P-table printa
J 4 DP$EXTRACT Make P-table printa
K 4 DP$EXTRACT Make P-table printa
L 4 DP$EXTRACT Make P-table printa
M 4 DP$EXTRACT Make P-table printa
N 4 DP$EXTRACT Make P-table printa
B 5 GET_LINE Condition getline
C 5 GET_LINE Condition getline
D 5 FORMAT_PROMPT Format and load
E 5 FORMAT_PROMPT Format and load
F 5 DSV$SHOWDEVICE Display device
G 5 DSV$SHOWDEVICE Display device
H 5 DSV$SHOWSELECT Display informa
I 5 DSV$SHOWSELECT Display informa
J 5 DSV$DEATTACH Deattach device
K 5 Deattach subtree
L 5 DSV$SELECT Select device for t
M 5 DSV$SELECT Select device for t
N 5 DSV$SELECT Select device for t
B 6 DSV$SELECT Select device for t
C 6 DSV$SELECT Select device for t
D 6 DSV$DESELECT Deselect device f
E 6 DSV$DESELECT Deselect device f
F 6 DSV$DESELECT Deselect device f
G 6 DSV$DESELECT Deselect device f
H 6 DSV$DESELECT Deselect device f
I 6 SHOW_DEV Routine
J 6 SHOW_DEV Routine
K 6 SHOW_DEV Routine
L 6 INI$PTABLE Initialize P-table
M 6 INI$PTABLE Initialize P-table
N 6 Check ambiguous device
B 7 Check ambiguous device
C 7 Locate /Adapter value
D 7 Locate /Adapter value
E 7 LOC$PTABLE Routine
F 7 LOC$PTABLE Routine
G 7 LOC$PTDESC Locate PT-descripto
H 7 LOC$PTDESC Locate PT-descripto
I 7 INS$PTABLE Insert P-table
J 7 INS$PTABLE Insert P-table
K 7 Symbol table
L 7 Symbol table
M 7 Symbol table
N 7 Symbol table
B 8 Psect synopsis
C 8 Cross reference
D 8 Cross reference
E 8 Cross reference
F 8 Cross reference
G 8 Cross reference
H 8 Cross reference
I 8 Cross reference
J 8 Cross reference
K 8 Cross reference
L 8 Cross reference
M 8 Cross reference
N 8 *** BOOTDRIVR non-interrupt I/
B 9 *** BOOTDRIVR non-interrupt I/
C 9 *** BOOTDRIVR non-interrupt I/
D 9 *** BOOTDRIVR non-interrupt I/
E 9 Declarations
F 9 Declarations
G 9 DRIVER FIXED DATA AREA
H 9 BOO$QIO - BOOTSTRAP QIO ROUTIN
I 9 BOO$QIO - BOOTSTRAP QIO ROUTIN
J 9 BOO$QIO - BOOTSTRAP QIO ROUTIN
K 9 BOO$QIO - BOOTSTRAP QIO ROUTIN
L 9 BOO$QIO - BOOTSTRAP QIO ROUTIN
M 9 BOO$MAP - ROUTINE TO MAP DATA
N 9 BOO$MAP - ROUTINE TO MAP DATA
B 10 BOO$PURDPR - Purge UBA Buffere
C 10 BOO$PURDPR - Purge UBA Buffere
D 10 BOO$PURDPR - Purge UBA Buffere
E 10 BOO$SELECT - Select boot drive
F 10 BOO$SELECT - Select boot drive
G 10 Symbol table
H 10 Psect synopsis
I 10 Cross reference
J 10 Cross reference
K 10 Cross reference
L 10 Cross reference
M 10 - BOOTSTRAP FILEREAD IO MODULE
N 10 - BOOTSTRAP FILEREAD IO MODULE
B 11 - BOOTSTRAP FILEREAD IO MODULE
C 11 - BOOTSTRAP FILEREAD IO MODULE
D 11 - BOOTSTRAP FILEREAD IO MODULE
E 11 RDWRTLBN - READ/WRITE LOGICAL
F 11 BOO$CACHE_INIT - INIT FILEREAD
G 11 BOO$CACHE_INIT - INIT FILEREAD
H 11 BOO$CACHE_INIT - INIT FILEREAD
I 11 BOO$IMAGE_ATT - Get image attr
J 11 SYS$ASSIGN, Dummy assign devic
K 11 Symbol table
L 11 Cross reference
M 11 Cross reference
N 11 *** BUFCTL QIO buffer control
B 12 *** BUFCTL QIO buffer control
C 12 *** BUFCTL QIO buffer control
D 12 GET ONE BYTE OF DATA FROM USER
E 12 PUT ONE BYTE OF DATA INTO USER
F 12 INITIALIZE FOR SINGLE BYTE TRA
G 12 MOVE FROM USER BUFFER
H 12 MOVE TO USER BUFFER
I 12 FILL SYSTEM PTE WITH TRANSFER
J 12 Symbol table
K 12 Psect synopsis
L 12 Cross reference
M 12 Cross reference
N 12 *** BUFFER GETBUF/RELBUF routi
B 13 *** BUFFER GETBUF/RELBUF routi
C 13 DECLARATIONS
D 13 GET A VERTUAL MEMORY BUFFER.
E 13 GET A VERTUAL MEMORY BUFFER.
F 13 RELEASE A MEMORY BUFFER.
G 13 RELEASE A MEMORY BUFFER.
H 13 Symbol table
I 13 Psect synopsis
J 13 Cross reference
K 13 Cross reference
L 13 Cross reference
M 13 Cross reference
N 13 *** CANCEL cancel QIO
B 14 *** CANCEL cancel QIO
C 14 *** CANCEL cancel QIO
D 14 CANCEL I/O ON CHANNEL
E 14 CANCEL I/O ON CHANNEL
F 14 CANCEL I/O ON CHANNEL
G 14 Symbol table
H 14 Symbol table
I 14 Psect synopsis
J 14 Cross reference
K 14 Cross reference
L 14 Cross reference

```
M 14   *** CHAN730 Channel services f
N 14   *** CHAN730 Channel services f
B 15   *** CHAN730 Channel services f
C 15   DECLARATIONS
D 15   DECLARATIONS
E 15   DECLARATIONS
F 15   DECLARATIONS
G 15   DECLARATIONS
H 15   $CHANNEL CALL
I 15   $CHANNEL CALL
J 15   $CHANNEL CALL - ADAPTER INIT F
K 15   $CHANNEL CALL - BUS INIT FUNCT
L 15   $CHANNEL CALL - ABORT FUNCTION
M 15   $CHANNEL CALL - ENABLE INTERRU
N 15   $CHANNEL CALL - PURGE
B 16   $CHANNEL CALL - DISABLE INTERR
C 16   $CHANNEL CALL - CLEAR ADAPTER
D 16   $CHANNEL CALL - GET ADAPTER ST
E 16   $CHANNEL CALL - SET DEFEAT PAR
F 16   $CHANNEL CALL - CLEAR DEFEAT P
G 16   $CHANNEL CALL - COMMON CALL RE
H 16   $SETMAP CALL
I 16   $SETMAP CALL
J 16   $SETMAP CALL
K 16   $SETMAP CALL
L 16   $SETMAP CALL
```

```
                                021D   916 ;+
                                021D   917 ; Routine to invalidate all adapter map entries
                                021D   918 ; Called by: Bsbw CLRMAP
                                021D   919 ; R6 = Adapter CSR address
                                021D   920 ;-
                                021D   921
                                021D   922 CLRMAP:
                          50    D4 021D   923         CLRL    R0                              ; Clear R0
                    0800 C640   D4 021F   924 10$:    CLRL    UBI$L_MAP(R6)[R0]               ; Clear the map entry
                F3  50  000001FF 8F F3 0224   925         AOBLEQ  #511, R0, 10$               ; Clear all 512 entries         [10]
                                05 022C   926         RSB                                     ; Return
                                022D   927
                                022D   928
                                022D   929 ;+
                                022D   930 ; ROUTINE TO SET ADAPTER VIRTUAL ADDRESS REGISTER
                                022D   931 ; CALLED BY BSBW SETVAR
                                022D   932 ; R2 = BASE ADDRESS
                                022D   933 ; R3 = START PHYSICAL ADDRESS
                                022D   934 ; R6 = ADAPTER CSR ADDRESS
                                022D   935 ; R10 = BYTE COUNT
                                022D   936 ;-
                                022D   937 SETVAR:
                                05 022D   938         RSB                                     ; RETURN
                                022E   939
                                022E   940
                                022E   941 ;+
                                022E   942 ; ROUTINE TO SET THE ADAPTER BYTE COUNT REGISTER
                                022E   943 ; CALLED BY: BSBW SETCNT
                                022E   944 ; R6 = ADAPTER CSR ADDRESS
                                022E   945 ; R10 = BYTE COUNT
                                022E   946 ;-
                                022E   947 SETCNT:
                                05 022E   948         RSB                                     ; RETURN
                                022F   949
                                022F   950 ;+
                                022F   951 ; ROUTINE TO VERIFY THAT THE BASE ADDRESS
                                022F   952 ; PLUS THE NUMBER OF PAGES REQUIRED WILL
                                022F   953 ; FIT WITHIN THE MAPPING SPACE
                                022F   954 ; R2 = MAP BASE
                                022F   955 ; R8 = TOTAL PAGES TO MAP
                                022F   956 ;-
                                022F   957 BASCHK:
          50    00660020 8F   D0 022F   958         MOVL    #DS$_PROGERR,R0                 ; NO - ERROR
              51    52    58  C1 0236   959         ADDL3   R8,R2,R1                        ; BASE + TOTAL PAGES
        00000200 8F   51    D1 023A   960         CMPL    R1, #512                        ; Is top register in range?     [10]
                      07    14 0241   961         BGTR    10$                             ; YES-
          50    00660001 8F   D0 0243   962         MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                                024A   963
                                05 024A   964 10$:    RSB                                     ; RETURN
                                024B   965
                                024B   966 ;+
                                024B   967 ; ROUTINE TO VALIDATE TRANSFER ADDRESS
                                024B   968 ; NOT IMPLEMENTED
                                024B   969 ;-
                                024B   970 RNGCHK:
                    50    01    D0 024B   971         MOVL    #1,R0                           ; Success
                                05 024E   972         RSB                                     ; RETURN
```

```
                                 024F   973
                                 024F   974 ;+
                                 024F   975 ; ROUTINE TO VERIFY THAT REQUESTED BYTE COUNT
                                 024F   976 ; WILL FIT IN REQUESTED BUFFER SIZE
                                 024F   977 ;-
                                 024F   978 CNTCHK:
                          0E     BB  024F   979          PUSHR   #^M<R1,R2,R3>                   ; SAVE R1 - R3
                       52   0C   BC  7D 0251   980          MOVQ    @STM$_PHY(AP),R2                ; BEGIN ADDR-> R2, END -> R3
                    53   01FF   8F  A8 0255   981          BISW2   #^X1FF,R3                       ; FORCE TO END OF PAGE
                         53   52   C2 025A   982          SUBL2   R2,R3                           ; FIND NUMBER OF BYTES
                              53   D6 025D   983          INCL    R3                              ; PLUS ONE
                       51   14   AC  D0 025F   984          MOVL    STM$_CNT(AP),R1                 ; COUNT TO R1
                 50 00660001   8F  D0 0263   985          MOVL    #DS$_NORMAL,R0                  ; INITIAL COMPLETION CODE
                         53   51   D1 026A   986          CMPL    R1,R3                           ; WILL BYTE COUNT FIT ?
                              07   15 026D   987          BLEQ    CNTCHK_RTN                      ; YES-
                 50 00660020   8F  D0 026F   988          MOVL    #DS$_PROGERR,R0                 ; NO -
                                 0276   989
                                 0276   990 CNTCHK_RTN:
                          0E     BA  0276   991          POPR    #^M<R1,R2,R3>                   ; RESTORE REGISTERS
                                 05  0278   992          RSB                                     ; RETURN
                                 0279   993
                                 0279   994 ;+
                                 0279   995 ;
                                 0279   996 ; SETMAP  "GLOBAL" EXIT
                                 0279   997 ;
                                 0279   998 ;-
                                 0279   999
                                 0279  1000 STM_NORMAL_RTN:
                 50 00660001   8F  D0 0279  1001          MOVL    #DS$_NORMAL,R0                  ; IT WORKS
                                 0280  1002
                                 0280  1003 STM_CALL_RTN:
                                 04  0280  1004          RET                                     ; RETURN TO CALLER
```

```
                        0281  1006          .SBTTL  DSP$SHOWMBA
                        0281  1007 ;++
                        0281  1008 ; FUNCTIONAL DESCRIPTION:
                        0281  1009 ;
                        0281  1010 ;       THIS ROUTINE WILL PRINT OUT THE CURRENT STATE OF OF THE MBA
                        0281  1011 ;       AS SPECIFIED BY THE ARGLIST
                        0281  1012 ;
                        0281  1013 ; CALLING SEQUENCE:
                        0281  1014 ;
                        0281  1015 ;       CALLS   #1,DSP$SHOWMBA OR
                        0281  1016 ;       CALLG   ARGLIST,DSP$SHOWMBA
                        0281  1017 ;
                        0281  1018 ; INPUT PARAMETERS:
                        0281  1019 ;
                        0281  1020 ;       4(AP)   -> ADDRESS OF MBA CSR
                        0281  1021 ;
                        0281  1022 ; IMPLICIT INPUTS:
                        0281  1023 ;
                        0281  1024 ;       NONE
                        0281  1025 ;
                        0281  1026 ; OUTPUT PARAMETERS:
                        0281  1027 ;
                        0281  1028 ;       NONE
                        0281  1029 ;
                        0281  1030 ; IMPLICIT OUTPUTS:
                        0281  1031 ;
                        0281  1032 ;       NONE
                        0281  1033 ;
                        0281  1034 ; COMPLETION CODES:
                        0281  1035 ;
                        0281  1036 ;       DS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                        0281  1037 ;
                        0281  1038 ; SIDE EFFECTS:
                        0281  1039 ;
                        0281  1040 ;       NONE
                        0281  1041 ;--
                        0281  1042
                   007C 0281  1043 .ENTRY  DSP$SHOWMBA,^M<R2,R3,R4,R5,R6>          ; ENTRY MASK (SAME AS SHOWCHAN)
50  006600B0 8F      D0 0283  1044          MOVL    #DS$_NOTIMP,R0
                     04 028A  1045          RET                                     ; RETURN
```

```
                                028B  1047          .SBTTL  DSP$SHOWUBI
                                028B  1048 ;++
                                028B  1049 ; FUNCTIONAL DESCRIPTION:
                                028B  1050 ;
                                028B  1051 ;       THIS PROCEEDURE WILL INTERPRET THE CONTENTS OF THE UBI
                                028B  1052 ;       POINTED TO BY THE ARGLIST
                                028B  1053 ;
                                028B  1054 ; CALLING SEQUENCE:
                                028B  1055 ;
                                028B  1056 ;       PUSHL   <CSR ADDR>
                                028B  1057 ;       CALLS   #1,DSP$SHOWUBI
                                028B  1058 ; OR    CALLG   <ARGLIST>,DSP$SHOWUBI
                                028B  1059 ;
                                028B  1060 ; INPUT PARAMETERS:
                                028B  1061 ;
                                028B  1062 ;       4(AP)   -> ADDRESS OF UBI CSR
                                028B  1063 ;
                                028B  1064 ; IMPLICIT INPUTS:
                                028B  1065 ;
                                028B  1066 ;       NONE
                                028B  1067 ;
                                028B  1068 ; OUTPUT PARAMETERS:
                                028B  1069 ;
                                028B  1070 ;       NONE
                                028B  1071 ;
                                028B  1072 ; IMPLICIT OUTPUTS:
                                028B  1073 ;
                                028B  1074 ;       NONE
                                028B  1075 ;
                                028B  1076 ; COMPLETION CODES:
                                028B  1077 ;
                                028B  1078 ;       DS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                                028B  1079 ;
                                028B  1080 ; SIDE EFFECTS:
                                028B  1081 ;
                                028B  1082 ;       NONE
                                028B  1083 ;--
                                028B  1084
                         007C   028B  1085 .ENTRY  DSP$SHOWUBI,^M<R2,R3,R4,R5,R6>          ; ENTRY MASK (SAME AS SHOWCHAN)
              56   04 AC     D0  028D  1086        MOVL    4(AP),R6                        ; COPY ADDRESS OF UBI CSR
                     002C     30  0291  1087        BSBW    SHOW_UBI                        ; INTERPRET THE UBI ADDRESS
        50   00660001 8F     D0  0294  1088        MOVL    #DS$_NORMAL,R0
                             04  029B  1089        RET                                     ; SUCCESS OF COURSE
```

```
                                029C  1091          .SBTTL  $SHOWCHAN CALL
                                029C  1092 ;++
                                029C  1093 ; FUNCTIONAL DESCRIPTION:
                                029C  1094 ;
                                029C  1095 ;       DISPLAY ADAPTER REGISTERS
                                029C  1096 ;
                                029C  1097 ; CALLING SEQUENCE:
                                029C  1098 ;
                                029C  1099 ;       CALLG   ARGLST, @#DS$SHOWCHAN
                                029C  1100 ;         OR
                                029C  1101 ;       CALLS   #1,@#DS$SHOWCHAN
                                029C  1102 ;
                                029C  1103 ; INPUT PARAMETERS:
                                029C  1104 ;
                                029C  1105 ;       4(AP) LOGICAL UNIT NUMBER
                                029C  1106 ;
                                029C  1107 ; IMPLICIT INPUTS:
                                029C  1108 ;
                                029C  1109 ;       NONE
                                029C  1110 ;
                                029C  1111 ; OUTPUT PARAMETERS:
                                029C  1112 ;
                                029C  1113 ;       R0      =       COMPLETION CODE
                                029C  1114 ;
                                029C  1115 ; IMPLICIT OUTPUTS:
                                029C  1116 ;
                                029C  1117 ;       NONE
                                029C  1118 ;
                                029C  1119 ; COMPLETION CODES:
                                029C  1120 ;
                                029C  1121 ;       DS$_NORMAL      =       SUCCESS
                                029C  1122 ;       DS$_PROGERR     =       PROGRAM ERROR
                                029C  1123 ;                                MISSING/INVALID P-TABLE
                                029C  1124 ;
                                029C  1125 ; SIDE EFFECTS:
                                029C  1126 ;
                                029C  1127 ;       NONE
                                029C  1128 ;--
                                029C  1129
                         007C   029C  1130 .ENTRY  DSX$SHOWCHAN,^M<R2,R3,R4,R5,R6>
                                029E  1131
          52    04 AC     D0    029E  1132          MOVL    SHW$_UNIT(AP),R2                ; GET THE LOGICAL UNIT NUMBER
                0065      30    02A2  1133          BSBW    BLDCDB                          ; BUILD THE CDB
                17 50     E9    02A5  1134          BLBC    R0, 20$                         ; Exit if error                 [09]
                                02A8  1135
                0015      30    02A8  1136 10$:     BSBW    SHOW_UBI                        ; INTERPRET STATUS FOR UBI
                                02AB  1137
                                02AB  1138          $DS_PRINTB_S    L^FMT_CRLF          ; SEPARATE THE SHOWCHAN         [07]
    00000000'EF     9F        02AB                        PUSHAB  L^FMT_CRLF
00000000'9F     01  FB        02B1                        CALLS   #$$N,@#DS$PRINTB
50  00660001 8F     D0        02B8  1139          MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                                02BF  1140
                         04     02BF  1141 20$:     RET                                     ; RETURN
```

```
                        02C0  1143          .SBTTL  SHOW_UBI STATUS
                        02C0  1144 ;++
                        02C0  1145 ; FUNCTIONAL DESCRIPTION:
                        02C0  1146 ;
                        02C0  1147 ;       SUBROUTINE TO PRINT UBI STATUS
                        02C0  1148 ;
                        02C0  1149 ; CALLING SEQUENCE:
                        02C0  1150 ;
                        02C0  1151 ;       BSBW    SHOW_UBI
                        02C0  1152 ;
                        02C0  1153 ; INPUT PARAMETERS:
                        02C0  1154 ;
                        02C0  1155 ;       R6=ADDRESS OF UBI CSR
                        02C0  1156 ;
                        02C0  1157 ; IMPLICIT INPUTS:      NONE
                        02C0  1158 ;
                        02C0  1159 ; OUTPUT PARAMETERS:    NONE
                        02C0  1160 ;
                        02C0  1161 ; IMPLICIT OUTPUTS:     NONE
                        02C0  1162 ;
                        02C0  1163 ; COMPLETION CODES:     NONE
                        02C0  1164 ;
                        02C0  1165 ; SIDE EFFECTS:         NONE
                        02C0  1166 ;--
                        02C0  1167
                        02C0  1168 SHOW_UBI:
                        02C0  1169          $DS_PRINTB_S    L^FMT_UBI_HDR           ; PRINT THE HEADER              [07]
    00000003'EF   9F    02C0                        PUSHAB  L^FMT_UBI_HDR
00000000'9F   01    FB  02C6                        CALLS   #$$N, @#DS$PRINTB
                    05  02CD  1170          RSB                                     ; RETURN
```

```
V7-ENSAA-7.0    SHOWBITS                                    *** CHAN730 Channel services for NEBULA  27-JUL-1984 15:04:35  VAX-11 Macro V03-01          Page  32
CHAN730                                                     SHOWBITS                                 23-MAY-1984 14:09:53  DMA1:[SYSO.SYSMAINT]CHAN730.MAR;4(23)
V6-11
```

```
                                02CE  1172          .SBTTL  SHOWBITS
                                02CE  1173 ;++
                                02CE  1174 ; FUNCTIONAL DESCRIPTION:
                                02CE  1175 ;
                                02CE  1176 ;       SUBROUTINE TO CVTREG AND PRINTB CSR BITS
                                02CE  1177 ;
                                02CE  1178 ; CALLING SEQUENCE:
                                02CE  1179 ;
                                02CE  1180 ;       BSBW    SHOWBITS
                                02CE  1181 ;
                                02CE  1182 ; INPUT PARAMETERS:
                                02CE  1183 ;
                                02CE  1184 ;       R2      ADDRESS OF FAO STRING TO BE OUTPUT
                                02CE  1185 ;       R3      ADDRESS OF DEVICE REGISTER
                                02CE  1186 ;       R4      ADDRESS OF ASCIC BIT DESCRIPTOR STRING
                                02CE  1187 ;
                                02CE  1188 ; IMPLICIT INPUTS:       NONE
                                02CE  1189 ;
                                02CE  1190 ; OUTPUT PARAMETERS:     NONE
                                02CE  1191 ;
                                02CE  1192 ; IMPLICIT OUTPUTS:      NONE
                                02CE  1193 ;
                                02CE  1194 ; COMPLETION CODES:      NONE
                                02CE  1195 ;
                                02CE  1196 ; SIDE EFFECTS:          NONE
                                02CE  1197 ;--
                                02CE  1198
                                02CE  1199 SHOWBITS:
        00000028'EF    63   D0  02CE  1200          MOVL    (R3), FMT_CVT_ARG+8             ; CONVERT DEVICE REGISTER BITS
        0000002C'EF    54   D0  02D5  1201          MOVL    R4,FMT_CVT_ARG+12               ; USING THIS MASK
                                02DC  1202          $DS_CVTREG_G  FMT_CVT_ARG               ; INTO ASCII IN SHWBUF
                                                                  CALLG   FMT_CVT_ARG, @#DS$CVTREG
00000000'9F     00000020'EF    FA  02DC
                00000068'EF    DF  02E7  1203          PUSHAL  SHWBUF                          ; PARAM 4 ADDRESS OF BIT ENCODINGS
                           63  DD  02ED  1204          PUSHL   (R3)                            ; PARAM 3 IS REG CONTENTS
    7E  53      C0000000   8F  CB  02EF  1205          BICL3   #3 @ 30, R3, -(SP)              ; PARAM 2 IS REGISTER ADDRESS
                           62  9F  02F7  1206          PUSHAB  (R2)                            ; PARAM 1 IS FAO ASCIC STRING
        00000000'9F        04  FB  02F9  1207          CALLS   #4,@#DS$PRINTB                  ; CALL OUTPUT ROUTINE
                               05  0300  1208          RSB                                     ; RETURN
```

```
                        0301  1210          .SBTTL  DETERMINE ADAPTER SPECIFIC STATUS
                        0301  1211 ;++
                        0301  1212 ; FUNCTIONAL DESCRIPTION:
                        0301  1213 ;
                        0301  1214 ;       DETERMINE ADAPTER SPECIFIC STATUS
                        0301  1215 ;
                        0301  1216 ; CALLING SEQUENCE:
                        0301  1217 ;
                        0301  1218 ;       BSBW    ADPSTS
                        0301  1219 ;
                        0301  1220 ; INPUT PARAMETERS.
                        0301  1221 ;
                        0301  1222 ;       R5      =       CHANNEL DATA BLOCK
                        0301  1223 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                        0301  1224 ;
                        0301  1225 ; IMPLICIT INPUTS:      NONE
                        0301  1226 ;
                        0301  1227 ; OUTPUT PARAMETERS:    NONE
                        0301  1228 ;
                        0301  1229 ; IMPLICIT OUTPUTS:     NONE
                        0301  1230 ;
                        0301  1231 ; COMPLETION CODES:     NONE
                        0301  1232 ;
                        0301  1233 ; SIDE EFFECTS:         NONE
                        0301  1234 ;--
                        0301  1235
                        0301  1236 ADPSTS:
50    D4                0301  1237          CLRL    R0                          ; CLEAR R0
      05                0303  1238          RSB                                 ; RETURN
```

```
                     0304  1240          .SBTTL  DETERMINE GENERAL ADAPTER STATUS
                     0304  1241 ;++
                     0304  1242 ; FUNCTIONAL DESCRIPTION:
                     0304  1243 ;
                     0304  1244 ;       DETERMINE GENERAL ADAPTER STATUS
                     0304  1245 ;
                     0304  1246 ; CALLING SEQUENCE:
                     0304  1247 ;
                     0304  1248 ;       BSBW    GENSTS
                     0304  1249 ;
                     0304  1250 ; INPUT PARAMETERS:
                     0304  1251 ;
                     0304  1252 ;       R5      =       CHANNEL DATA BLOCK
                     0304  1253 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0304  1254 ;
                     0304  1255 ; IMPLICIT INPUTS:      NONE
                     0304  1256 ;
                     0304  1257 ; OUTPUT PARAMETERS:    NONE
                     0304  1258 ;
                     0304  1259 ; IMPLICIT OUTPUTS:     NONE
                     0304  1260 ;
                     0304  1261 ; COMPLETION CODES:     NONE
                     0304  1262 ;
                     0304  1263 ; SIDE EFFECTS:         NONE
                     0304  1264 ;--
                     0304  1265
                     0304  1266 GENSTS:
50   D4              0304  1267          CLRL    R0                          ; CLEAR R0
     05              0306  1268          RSB                                 ; RETURN
```

```
                                0307  1270          .SBTTL  DETERMINE ADAPTER HARDWARE ERROR STATUS
                                0307  1271 ;++
                                0307  1272 ; FUNCTIONAL DESCRIPTION:
                                0307  1273 ;
                                0307  1274 ;       DETERMINE ADAPTER HARDWARE ERROR STATUS
                                0307  1275 ;
                                0307  1276 ; CALLING SEQUENCE:
                                0307  1277 ;
                                0307  1278 ;       BSBW    HDWSTS
                                0307  1279 ;
                                0307  1280 ; INPUT PARAMETERS:
                                0307  1281 ;
                                0307  1282 ;       R5      =       CHANNEL DATA BLOCK
                                0307  1283 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                0307  1284 ;
                                0307  1285 ; IMPLICIT INPUTS:      NONE
                                0307  1286 ;
                                0307  1287 ; OUTPUT PARAMETERS:    NONE
                                0307  1288 ;
                                0307  1289 ; IMPLICIT OUTPUTS:     NONE
                                0307  1290 ;
                                0307  1291 ; COMPLETION CODES:     NONE
                                0307  1292 ;
                                0307  1293 ; SIDE EFFECTS:        NONE
                                0307  1294 ;--
                                0307  1295
                                0307  1296 HDWSTS:
50   D4                         0307  1297          CLRL    R0                      ; CLEAR R0
     05                         0309  1298          RSB                             ; RETURN
```

```
                        030A  1300
                        030A  1301          .SBTTL  BUILD THE CHANNEL DATA BLOCK
                        030A  1302 ;++
                        030A  1303 ; FUNCTIONAL DESCRIPTION:
                        030A  1304 ;
                        030A  1305 ; ROUTINE TO BUILD A CHANNEL DATA BLOCK FOR
                        030A  1306 ; LOGICAL UNIT NUMBER FOUND IN R2
                        030A  1307 ;
                        030A  1308 ; CALLING SEQUENCE:
                        030A  1309 ;
                        030A  1310 ;       BSBW    BLDCDB
                        030A  1311 ;
                        030A  1312 ;
                        030A  1313 ; INPUT PARAMETERS:
                        030A  1314 ;
                        030A  1315 ;       R2      =       LOGICAL UNIT NUMBER
                        030A  1316 ;
                        030A  1317 ; IMPLICIT INPUTS:      NONE
                        030A  1318 ;
                        030A  1319 ; OUTPUT PARAMETERS:
                        030A  1320 ;
                        030A  1321 ;       R0      =       COMPLETION CODE
                        030A  1322 ;       R5      =       CDB ADDRESS
                        030A  1323 ;       R6      =       ADAPTER ADDRESS
                        030A  1324 ;
                        030A  1325 ; IMPLICIT OUTPUTS:     NONE
                        030A  1326 ;
                        030A  1327 ; COMPLETION CODES:
                        030A  1328 ;
                        030A  1329 ;       DS$_NORMAL      =       SUCCESS
                        030A  1330 ;       DS$_ERROR       =       ERROR ($GPHARD)
                        030A  1331 ;                                 MISSING/INVALID P-TABLE
                        030A  1332 ;
                        030A  1333 ; SIDE EFFECTS:
                        030A  1334 ;
                        030A  1335 ;       DESTROYS R5, R6
                        030A  1336 ;       SETS FLAG (CDB$M_UBI) TO INDICATE ADAPTER TYPE
                        030A  1337 ;
                        030A  1338 ;--
                        030A  1339
                        030A  1340 BLDCDB:
                        030A  1341          $DS_GPHARD_S    R2,-(SP)                ; GET THE HARDWARE P-TABLE
                     7E DF  030A                    PUSHAL  -(SP)
                     52 DD  030C                    PUSHL   R2
00000000'9F          02 FB  030E                    CALLS   #2, @#DS$GPHARD
           56        8E D0  0315  1342              MOVL    (SP)+,R6                ; GET ADDRESS OF HP-TABLE
              2C     50 E9  0318  1343              BLBC    R0, BLD_RTN             ; IS THERE ONE ?                    [05]
55    00000000'EF     DE  0318  1344              MOVAL   CDB$BLOCK,R5            ; CDB TO R5
65    FFFFFFDF 8F     CA  0322  1345              BICL    #^CCDB$M_IE,CDB$L_FLAGS(R5)     ; Clear flags
                      0329  1346
              20 A6   D5  0329  1347 10$:        TSTL    HP$A_LINK(R6)           ; SEARCH FOR ADAPTER ENTRY
                 06   13  032C  1348              BEQL    20$                     ; FOUND IT
           56 20 A6   D0  032E  1349              MOVL    HP$A_LINK(R6),R6        ; KEEP LOOKING
                 F5   11  0332  1350              BRB     10$                     ;
                      0334  1351 20$:
        04 A5    56    D0  0334  1352              MOVL    R6,CDB$A_TBL(R5)        ; SAVE THE P-TABLE ADDRESS
           56 18 A6    D0  0338  1353              MOVL    HP$A_DEVICE(R6),R6      ; Base of CSR's
```

```
                                 033C  1354
                00 65    U2   E3  033C  1355          BBCS    #CDB$V_UBI,CDB$L_FLAGS(R5),40$  ; Flag as an UBI
                                 0340  1356
          50    00660001 8F   D0  0340  1357 40$:     MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                                 0347  1358
                                 0347  1359 BLD_RTN:
                              05 0347  1360          RSB                                     ; RETURN
```

```
                     0348  1362          .SBTTL  SET ADAPTER VECTOR(S)
                     0348  1363 ;++
                     0348  1364 ; FUNCTIONAL DESCRIPTION:
                     0348  1365 ;
                     0348  1366 ;       SET SCB VECTOR FOR ADAPTER/DEVICE TO POINT TO CHANNEL SERVICE
                     0348  1367 ;
                     0348  1368 ; CALLING SEQUENCE:
                     0348  1369 ;
                     0348  1370 ;       BSBW    SETVEC
                     0348  1371 ;
                     0348  1372 ; INPUT PARAMETERS:
                     0348  1373 ;
                     0348  1374 ;       R5      =       CHANNEL DATA BLOCK
                     0348  1375 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0348  1376 ;
                     0348  1377 ; IMPLICIT INPUTS:      NONE
                     0348  1378 ;
                     0348  1379 ; OUTPUT PARAMETERS:    NONE
                     0348  1380 ;
                     0348  1381 ; IMPLICIT OUTPUTS:     NONE
                     0348  1382 ;
                     0348  1383 ; COMPLETION CODES:
                     0348  1384 ;
                     0348  1385 ;       DS$_NORMAL      =       SUCCESS
                     0348  1386 ;       DS$_IVVECT      =       INVALID VECTOR ($SETVEC)
                     0348  1387 ;       DS$_IVADDR      =       INVALID ADDRESS ($SETVEC)
                     0348  1388 ;
                     0348  1389 ; SIDE EFFECTS:         NONE
                     0348  1390 ;--
                     0348  1391
                     0348  1392 SETVEC:
                  04 BB   0348  1393          PUSHR   #^M<R2>                         ; SAVE A WORKING REGISTER
                     034A  1394
               51 04 A5 D0   034A  1395          MOVL    CDB$A_TBL(R5),R1                ; GET THE HARDWARE P-TABLE
            10 A5 24 A1 3C   034E  1396          MOVZWL  HP$W_VECTOR(R1),CDB$A_ADDR(R5)  ; SAVE CHANNEL VECTOR
                  65 20 C8   0353  1397          BISL    #CDB$M_IE,CDB$L_FLAGS(R5)       ; INDICATE INTERRUPTS ENABLED
                     0356  1398
51  00000200'EF 7E   0356  1399          MOVAQ   SCB_BASE+512,R1                 ; Point to SCB
52  00000200'EF 9E   035D  1400          MOVAB   SCB_UNKINT+512,R2               ; Point to image
            50 0100 8F 3C   0364  1401          MOVZWL  #2*<512/4>,R0                   ; Number of longwords to fill
                     0369  1402
                  81 82 DE   0369  1403 10$:    MOVAL   (R2)+,(R1)+                     ; Do one SCB page
                     FA 50 F5   036C  1404          SOBGTR  R0,10$                  ; Copy whole page
50  00660001 8F DO   036F  1405          MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                  04 BA   0376  1406          POPR    #^M<R2>                         ; RESTORE REGISTERS
                     05 0378  1407          RSB
```

```
                     0379  1409          .SBTTL  CLEAR ADAPTER VECTOR(S)
                     0379  1410 ;++
                     0379  1411 ; FUNCTIONAL DESCRIPTION:
                     0379  1412 ;
                     0379  1413 ;       CLEAR CHANNEL SERVICE POINTERS IN SCB VECTOR SPACE
                     0379  1414 ;
                     0379  1415 ; CALLING SEQUENCE:
                     0379  1416 ;
                     0379  1417 ;       BSBW    CLRVEC
                     0379  1418 ;
                     0379  1419 ; INPUT PARAMETERS:
                     0379  1420 ;
                     0379  1421 ;       R5      =       CHANNEL DATA BLOCK
                     0379  1422 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0379  1423 ;
                     0379  1424 ; IMPLICIT INPUTS:
                     0379  1425 ;
                     0379  1426 ;       NONE
                     0379  1427 ;
                     0379  1428 ; OUTPUT PARAMETERS:
                     0379  1429 ;
                     0379  1430 ;       R0      =       COMPLETION CODE
                     0379  1431 ;
                     0379  1432 ; IMPLICIT OUTPUTS:
                     0379  1433 ;
                     0379  1434 ;       NONE
                     0379  1435 ;
                     0379  1436 ; COMPLETION CODES:
                     0379  1437 ;
                     0379  1438 ;       DS$_NORMAL      =       SUCCESS
                     0379  1439 ;       DS$_IVVECT      =       INVALID VECTOR ($CLRVEC)
                     0379  1440 ;
                     0379  1441 ; SIDE EFFECTS:
                     0379  1442 ;
                     0379  1443 ;       NONE
                     0379  1444 ;--
                     0379  1445
                     0379  1446 CLRVEC:
65   20   CA         0379  1447          BICL    #CDB$M_IE,CDB$L_FLAGS(R5)       ; CLEAR IE
     05              037C  1448          RSB                                     ; RETURN WITH COMPLETION CODE
```

```
                                    1450          .SBTTL  INTERRUPT PRE-PROCESS
                                    1451 ;++
                                    1452 ; FUNCTIONAL DESCRIPTION:
                                    1453 ;
                                    1454 ;       THIS ROUTINE WILL PRE-PROCESS ALL CHANNEL ADAPTER INTERRUPTS
                                    1455 ;       AND PASS CONTROL TO THE DIAGNOSTIC PROGRAM
                                    1456 ;
                                    1457 ; CALLING SEQUENCE:
                                    1458 ;
                                    1459 ;       INTERRUPT
                                    1460 ;
                                    1461 ; INPUT PARAMETERS:
                                    1462 ;
                                    1463 ;       NONE
                                    1464 ;
                                    1465 ; IMPLICIT INPUTS:
                                    1466 ;
                                    1467 ;       CDB$A_VEC MUST HAVE BEEN PREVIOUSLY SET BY $CHANNEL CALL TO
                                    1468 ;       ENABLE INTERRUPTS
                                    1469 ;
                                    1470 ; OUTPUT PARAMETERS:
                                    1471 ;
                                    1472 ;       THE CHANNEL STATUS QUADWORD IS GENERATED
                                    1473 ;
                                    1474 ; IMPLICIT OUTPUTS:
                                    1475 ;
                                    1476 ;       NONE
                                    1477 ;
                                    1478 ; COMPLETION CODES:
                                    1479 ;
                                    1480 ;       NONE
                                    1481 ;
                                    1482 ; SIDE EFFECTS:
                                    1483 ;
                                    1484 ;--
```

```
                                037D  1486          .ALIGN  LONG
                                0380  1487 DSI$CHANNEL_VEC::
        06 00000000'EF    05    E0  0380  1488          BBS     #CDB$V_IE,CDB$L_FLAGS+CDB$BLOCK, -
                                0388  1489                          DSI$CHANNEL_COM         ; IS INTERRUPT EXPECTED
                                05    0388  1490          RSB                             ; NO - RETURN TO UNEXPECTED HANDLER
                                      0389  1491
                                      0389  1492          .ALIGN  LONG
                                      038C  1493 DSI$CHANNEL:
                             7E   7C  038C  1494          CLRQ    -(SP)                   ; Dummy space to match UBI
                                      038E  1495 DSI$CHANNEL_COM:
                       0063  8F   BB  038E  1496          PUSHR   #^M<R0,R1,R5,R6>        ; SAVE REGISTERS
                                      0392  1497                                          ; SAVE INTERRUPT LEVEL OF DEVICE
                                      0392  1498          DSBINT  #^X17                   ; & DISABLE DEVICE INTERRUPTS
                       7E    12   DB  0392                        MFPR    S^#PR$_IPL,-(SP)
                       12    17   DA  0395                        MTPR    #^X17,S^#PR$_IPL
                                      0398  1499
              55  00000000'EF   DE  0398  1500          MOVAL   CDB$BLOCK,R5            ; CDB ADDRESS TO R5
                   56   04 A5   D0  039F  1501          MOVL    CDB$A_TBL(R5),R6        ; P-TABLE ADDRESS TO R6
                   56   18 A6   D0  03A3  1502          MOVL    HP$A_DEVICE(R6),R6      ; CHANNEL BASE ADDRESS TO R6
                                      03A7  1503
                        FF5D    30  03A7  1504          BSBW    HDWSTS                  ; GET HARDWARE ERROR STATUS
                      51   50   D0  03AA  1505          MOVL    R0,R1                   ; SAVE THE STATUS
                        FF51    30  03AD  1506          BSBW    ADPSTS                  ; GET ADAPTER SPECIFIC STATUS
                      50   51   C8  03B0  1507          BISL    R1,R0                   ; COMBINE STATUS
                   14 A5   50   D0  03B3  1508          MOVL    R0,CDB$L_ST1(R5)        ; SAVE IT
                        FF4A    30  03B7  1509          BSBW    GENSTS                  ; GET GENERAL STATUS
                   18 A5   50   B0  03BA  1510          MOVW    R0,CDB$W_ST2(R5)        ; SAVE IT
                                03BE  1511
          18 A5    05   02   8E   F0  03BE  1512          INSV    (SP)+,#CHI$V_IPL,#CHI$S_IPL, -  ; STORE THE IPL OF THE INTERRUPT
                                03C4  1513                    CDB$W_ST2(R5)
                                03C4  1514
      1A A5   14 AE    FE00 8F   AB  03C4  1515          BICW3   #^C^X1FF,20(SP),CDB$W_RVR(R5)  ; SAVE THE VECTOR
                   18 A5    02   A8  03CC  1516          BISW    #CHI$M_DEVINT,CDB$W_ST2(R5)  ; FLAG AS DEVICE INTERRUPT
                                03D0  1517
                   50   0C A5   D0  03D0  1518          MOVL    CDB$A_STORE(R5),R0      ; HANDLER'S STORE AREA TO R0
                   80   14 A5   D0  03D4  1519          MOVL    CDB$L_ST1(R5),(R0)+     ; HARDWARE ERROR STATUS
                   60   18 A5   D0  03D8  1520          MOVL    CDB$W_ST2(R5),(R0)      ; RVR + GENERAL STATUS
                                03DC  1521
                14 AE   08 A5   D0  03DC  1522          MOVL    CDB$A_VEC(R5),20(SP)    ; BUILD THE HANDLER ADDRESS
                                03E1  1523
                                03E1  1524 INT_DSP:
                       0063  8F   BA  03E1  1525          POPR    #^M<R0,R1,R5,R6>        ; RESTORE R0, R1, R5, R6
                             8E   D5  03E5  1526          TSTL    (SP)+                   ; Remove JSB return
                                05    03E7  1527          RSB                             ; Dispatch to handler
                                03E8  1528
                                03E8  1529
                                03E8  1530          .END
```

```
$$ARGS          =  00000001     D
$$N             =  00000001     D
$$T1            =  00000008     D
$ER             =  00000001     D
$MODULE            00000060 R   D   02
ADPSTS             00000301 R   D   04
BASCHK             0000022F R   D   04
BIT...          =  0000001E     D
BIT0            =  00000001     D
BIT1            =  00000002     D
BIT10           =  00000400     D
BIT11           =  00000800     D
BIT12           =  00001000     D
BIT13           =  00002000     D
BIT14           =  00004000     D
BIT15           =  00008000     D
BIT16           =  00010000     D
BIT17           =  00020000     D
BIT18           =  00040000     D
BIT19           =  00080000     D
BIT2            =  00000004     D
BIT20           =  00100000     D
BIT21           =  00200000     D
BIT22           =  00400000     D
BIT23           =  00800000     D
BIT24           =  01000000     D
BIT25           =  02000000     D
BIT26           =  04000000     D
BIT27           =  08000000     D
BIT28           =  10000000     D
BIT29           =  20000000     D
BIT3            =  00000008     D
BIT30           =  40000000     D
BIT31           =  80000000     D
BIT4            =  00000010     D
BIT5            =  00000020     D
BIT6            =  00000040     D
BIT7            =  00000080     D
BIT8            =  00000100     D
BIT9            =  00000200     D
BLDCDB             0000030A R   D   04
BLD_RTN            00000347 R   D   04
CDB$A_ADDR         00000010     D
CDB$A_STORE        0000000C     D
CDB$A_TBL          00000004     D
CDB$A_VEC          00000008     D
CDB$BLOCK          00000000 RG  D   02
CDB$L_FLAGS        00000000     D
CDB$L_PTH          0000001C     D
CDB$L_ST1          00000014     D
CDB$M_IE        =  00000020     D
CDB$M_MBA       =  00000001     D
CDB$M_OFFSET    =  00000008     D
CDB$M_REV       =  00000010     D
CDB$M_UBA       =  00000002     D
CDB$M_UBI       =  00000004     D
CDB$V_IE        =  00000005     D
CDB$V_MBA       =  00000000     D
CDB$V_OFFSET    =  00000003     D
CDB$V_REV       =  00000004     D
CDB$V_UBA       =  00000001     D
CDB$V_UBI       =  00000002     D
CDB$W_RVR          0000001A     D
CDB$W_ST2          00000018     D
CDBSIZ          =  00000020     D
CHC$_ABORT      =  00000002     D
CHC$_CLEAR      =  00000006     D
CHC$_CLRDFT     =  00000009     D
CHC$_DSINT      =  00000005     D
CHC$_ENINT      =  00000004     D
CHC$_INITA      =  00000000     D
CHC$_INITB      =  00000001     D
CHC$_PURGE      =  00000003     D
CHC$_SELF_TEST  =  0000000B     D
CHC$_SETDFT     =  00000008     D
CHC$_STATUS     =  00000007     D
CHC$_STOP       =  0000000A     D
CHI$M_CHNINT    =  00000001     D
CHI$M_DEVINT    =  00000002     D
CHI$M_IPL       =  0000007C     D
CHI$S_IPL       =  00000005     D
CHI$V_CHNINT    =  00000000     D
CHI$V_DEVINT    =  00000001     D
CHI$V_IPL       =  00000002     D
CHM$_FORWARD    =  00000000     D
CHM$_INVALIDATE =  00000001     D
CHM$_MAP        =  00000002     D
CHM$_MFWDN      =  00000002     D
CHM$_MFWDNO     =  0000000A     D
CHM$_MFWDV      =  00000003     D
CHM$_MFWDVO     =  0000000B     D
CHM$_MREVN      =  00000006     D
CHM$_MREVNO     =  0000000E     D
CHM$_MREVV      =  00000007     D
CHM$_MREVVO     =  0000000F     D
CHM$_NFWDN      =  00000000     D
CHM$_NREVN      =  00000004     D
CHM$_OFFSET     =  00000008     D
CHM$_REVERSE    =  00000004     D
CHN$_ADR1       =  0000000C     D
CHN$_ADR2       =  00000010     D
CHN$_FUNC       =  00000008     D
CHN$_NARGS      =  00000004     D
CHN$_UNIT       =  00000004     D
CHN_ABORT          0000004E R   D   04
CHN_CALL_RTN       000000B0 R   D   04
CHN_CLEAR          00000084 R   D   04
CHN_CLRDFT         000000A0 R   D   04
CHN_DSINT          0000007F R   D   04
CHN_DSP            0000000F R   D   04
CHN_ENINT          00000057 R   D   04
CHN_INITA          00000035 R   D   04
CHN_INITB          00000041 R   D   04
CHN_NORMAL_RTN     000000A9 R   D   04
CHN_PURGE          00000076 R   D   04
CHN_SETDFT         00000097 R   D   04
CHN_STATUS         0000008D R   D   04
CHS$M_BADBDP    =  20000000     D
CHS$M_BUSIC     =  00800000     D
CHS$M_BUSINIT   =  00400000     D
CHS$M_BUSNXM    =  08000000     D
CHS$M_BUSPDN    =  01000000     D
CHS$M_CHNDPE    =  00000040     D
CHS$M_CHNERR    =  00000002     D
CHS$M_CHNMPE    =  00000080     D
CHS$M_CHPFOT    =  00000100     D
CHS$M_DEVBUS    =  00000010     D
CHS$M_DEVERR    =  00000004     D
CHS$M_DEVTO     =  00000020     D
CHS$M_ERRANY    =  0000000F     D
CHS$M_MBAATN    =  00100000     D
CHS$M_MBACPE    =  00200000     D
CHS$M_MBADTB    =  00040000     D
CHS$M_MBADTC    =  00080000     D
CHS$M_MBAEXC    =  00010000     D
CHS$M_MBANED    =  00020000     D
CHS$M_MBAWCKLWR =  02000000     D
CHS$M_MBAWCKUPR =  04000000     D
CHS$M_PGMERR    =  00000008     D
CHS$M_PGMHDE    =  00000800     D
CHS$M_SYSERR    =  00000001     D
CHS$M_SYSMEM    =  00000200     D
CHS$M_SYSSBI    =  00000400     D
CHS$M_UIE       =  10000000     D
CHS$V_BADBDP    =  0000001D     D
CHS$V_BUSIC     =  00000017     D
CHS$V_BUSINIT   =  00000016     D
CHS$V_BUSNXM    =  0000001B     D
CHS$V_BUSPDN    =  00000018     D
CHS$V_CHNDPE    =  00000006     D
CHS$V_CHNERR    =  00000001     D
CHS$V_CHNMPE    =  00000007     D
CHS$V_CHPFOT    =  00000008     D
CHS$V_DEVBUS    =  00000004     D
CHS$V_DEVERR    =  00000002     D
CHS$V_DEVTO     =  00000005     D
CHS$V_MBAATN    =  00000014     D
CHS$V_MBACPE    =  00000015     D
CHS$V_MBADTB    =  00000012     D
CHS$V_MBADTC    =  00000013     D
CHS$V_MBAEXC    =  00000010     D
CHS$V_MBANED    =  00000011     D
CHS$V_MBAWCKLWR =  00000019     D
CHS$V_MBAWCKUPR =  0000001A     D
CHS$V_PGMERR    =  00000003     D
CHS$V_PGMHDE    =  0000000B     D
CHS$V_SYSERR    =  00000000     D
CHS$V_SYSMEM    =  00000009     D
CHS$V_SYSSBI    =  0000000A     D
CHS$V_UIE       =  0000001C     D
CLRMAP             0000021D R   D   04
```

```
CLRVEC          00000379 R   D  04      DSI$CHANNEL     0000038C R   D  04      SCB_BASE        ********    X   04
CNTCHK          0000024F R   D  04      DSI$CHANNEL_COM 0000038E R   D  04      SCB_UNKINT      ********    X   04
CNTCHK_RTN      00000276 R   D  04      DSI$CHANNEL_VEC 00000380 RG  D  04      SETCNT          0000022E R   D  04
CVT$T_MBAMAP    00000054 RG  D  02      DSP$SHOWMBA     00000281 RG  D  04      SETMAP          000001E9 R   D  04
CVT$T_MBASR     00000050 RG  D  02      DSP$SHOWUBI     0000028B RG  D  04      SETVAR          0000022D R   D  04
CVT$T_UBADPR    00000058 RG  D  02      DSX$CHANNEL     00000000 RG  D  04      SETVEC          00000348 R   D  04
CVT$T_UBAMAP    0000005C RG  D  02      DSX$SETMAP      000000B1 RG  D  04      SHOWBITS        000002CE R   D  04
DS$CVTREG       ********     X  04      DSX$SHOWCHAN    0000029C RG  D  04      SHOW_UBI        000002C0 R   D  04
DS$GPHARD       ********     X  04      FMT_CRLF        00000000 R   D  03      SHW$K_BUFSIZ  =  00000080      D
DS$K_ERROR    = 00000002       D        FMT_CVT_ARG     00000020 R   D  02      SHW$_NARGS    =  00000001      D
DS$K_NORMAL   = 00000001       D        FMT_UBI_CSR     0000001F R   D  03      SHW$_UNIT     =  00000004      D
DS$K_SEVERE   = 00000004       D        FMT_UBI_HDR     00000003 R   D  03      SHWBUF          00000068 R   D  02
DS$K_SUBSYS   = 00000066       D        GENSTS          00000304 R   D  04      SIZ...        =  00000001      D
DS$K_WARNING  = 00000000       D        HDWSTS          00000307 R   D  04      STM$_BAS      =  00000010      D
DS$PRINTB       ********     X  04      HP$A_DEPENDENT  00000032         D      STM$_CNT      =  00000014      D
DS$_ARITH     = 00660DD0       D        HP$A_DEVICE     00000018         D      STM$_FUNC     =  00000008      D
DS$_ASBE      = 00660118       D        HP$A_DVA        0000001C         D      STM$_NARGS    =  00000006      D
DS$_BADLINK   = 006600F0       D        HP$A_LINK       00000020         D      STM$_PHY      =  0000000C      D
DS$_BADTYPE   = 006600E8       D        HP$B_DRIVE      0000000B         D      STM$_PTH      =  00000018      D
DS$_BIIC      = 00660120       D        HP$B_FLAGS      0000000A         D      STM$_UNIT     =  00000004      D
DS$_CHME      = 006600A8       D        HP$Q_DEVICE     00000000         D      STM_CALL_RTN    00000280 R   D  04
DS$_CHMK      = 006600E0       D        HP$T_DEVICE     0000000C         D      STM_NORMAL_RTN  00000279 R   D  04
DS$_DEVNAME   = 00660108       D        HP$T_TYPE       00000026         D      UBA$S_MAP_ADDR =  00000015      D
DS$_ERROR     = 00660002       D        HP$W_SIZE       00000008         D      UBA$V_MAP_ADDR =  00000000      D
DS$_FHWE      = 00660068       D        HP$W_VECTOR     00000024         D      UBA_DPR_BIT     0000003D R   D  03
DS$_FRAGBUF   = 00660080       D        INT_DSP         000003E1 R   D  04      UBA_MAP_BIT     0000003E R   D  03
DS$_ICBUSY    = 006600C8       D        INVALID         0000012B R   D  04      UBI$L_CSR     =  00000000      D
DS$_ICERR     = 006600C0       D        INVALIDATE      00000135 R   D  04      UBI$L_MAP     =  00000800      D
DS$_IHWE      = 00660060       D        JMPADR          000000E9 R   D  02      UBI$M_MAP_BO  =  02000000      D
DS$_ILLCHAR   = 00660018       D        MBA_MAP_BIT     0000003D R   D  03      UBI$M_MAP_VALID= 80000000      D
DS$_ILLPAGCNT = 00660078       D        MBA_SR_BIT      0000003D R   D  03      UBI$S_MAP_BO  =  00000001      D
DS$_ILLUNIT   = 00660100       D        MFWDN           00000150 R   D  04      UBI$S_MAP_DPD =  00000002      D
DS$_INSFMEM   = 00660050       D        MFWDNO          000001B6 R   D  04      UBI$S_MAP_PFN =  0000000F      D
DS$_IPL2HI    = 006600B8       D        MFWDV           0000013E R   D  04      UBI$V_MAP_BO  =  00000019      D
DS$_IVADDR    = 00660040       D        MFWDVO          000001A1 R   D  04      UBI$V_MAP_DPD =  00000015      D
DS$_IVVECT    = 00660038       D        MREVN           00000180 R   D  04      UBI$V_MAP_PFN =  00000000      D
DS$_KRNLSTK   = 00660090       D        MREVNO          000001DA R   D  04      UBI$V_MAP_VALID= 0000001F      D
DS$_LOGIC     = 00660070       D        MREVV           0000016B R   D  04      UBIVEC          000000ED R   D  02
DS$_MCHK      = 00660088       D        MREVVO          000001C8 R   D  04
DS$_MMOFF     = 00660058       D        NFWDN           0000015F R   D  04
DS$_NEEDUNIT  = 006600F8       D        NREVN           00000192 R   D  04
DS$_NODE      = 00660128       D        PAR$M_ATDEF   =  00000010        D
DS$_NOPCS     = 00660110       D        PAR$M_ATHI    =  00000008        D
DS$_NORMAL    = 00660001       D        PAR$M_ATLO    =  00000004        D
DS$_NOSUPPORT = 006600B1       D        PAR$M_NODEF   =  00000002        D
DS$_NOTDON    = 00660030       D        PAR$V_ATDEF   =  00000004        D
DS$_NOTIMP    = 006600B0       D        PAR$V_ATHI    =  00000003        D
DS$_NULLSTR   = 00660010       D        PAR$V_ATLO    =  00000002        D
DS$_OVERFLOW  = 00660008       D        PAR$V_NODEF   =  00000001        D
DS$_POWER     = 00660098       D        PAR$_BIN      =  00000002        D
DS$_PROGERR   = 00660020       D        PAR$_DEC      =  0000000A        D
DS$_SEVERE    = 00660004       D        PAR$_HEX      =  00000010        D
DS$_TRANSL    = 006600A0       D        PAR$_NO       =  00000000        D
DS$_TRUNCATE  = 00660028       D        PAR$_OCT      =  00000008        D
DS$_UNEXPINT  = 006600D8       D        PAR$_YES      =  00000001        D
DS$_VASFULL   = 00660048       D        PR$_IPL       =  00000012        D
DS$_WARNING   = 00660000       D        RNGCHK          0000024B R   D  04
```

Psect synopsis

PSECT name	Allocation		PSECT No.		Attributes									
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC BYTE
ABS	00000032	(50.)	01	(1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
WORK	000000F1	(241.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	NOVEC LONG
DATA	00000059	(89.)	03	(3.)	NOPIC	USR	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	NOVEC LONG
CODE	000003E8	(1000.)	04	(4.)	NOPIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFFRENCES...							
$$ARGS	=00000001	132 (2)	129 (2)	131 (2)	132 (2)					
$$N	=00000001	1169 (22)	#-1138 (21)	#-1169 (22)						
$$T1	=00000008	132 (2)	129 (2)	131 (2)	132 (2)					
$ER	=00000001	200 (3)								
$MODULE	00000060-R	200 (3)								
ADPSTS	00000301-R	1236 (24)	#-1506 (31)							
BASCHK	0000022F-R	957 (18)	#-769 (18)							
BIT...	=0000001E	128 (2)	104 (2)	125 (2)	126 (2)	128 (2)				
BIT0	=00000001	124 (2)								
BIT1	=00000002	124 (2)								
BIT10	=00000400	124 (2)								
BIT11	=00000800	124 (2)								
BIT12	=00001000	124 (2)								
BIT13	=00002000	124 (2)								
BIT14	=00004000	124 (2)								
BIT15	=00008000	124 (2)								
BIT16	=00010000	124 (2)								
BIT17	=00020000	124 (2)								
BIT18	=00040000	124 (2)								
BIT19	=00080000	124 (2)								
BIT2	=00000004	124 (2)								
BIT20	=00100000	124 (2)								
BIT21	=00200000	124 (2)								
BIT22	=00400000	124 (2)								
BIT23	=00800000	124 (2)								
BIT24	=01000000	124 (2)								
BIT25	=02000000	124 (2)								
BIT26	=04000000	124 (2)								
BIT27	=08000000	124 (2)								
BIT28	=10000000	124 (2)								
BIT29	=20000000	124 (2)								
BIT3	=00000008	124 (2)								
BIT30	=40000000	124 (2)								
BIT31	=80000000	124 (2)								
BIT4	=00000010	124 (2)								
BIT5	=00000020	124 (2)								
BIT6	=00000040	124 (2)								
BIT7	=00000080	124 (2)								
BIT8	=00000100	124 (2)								
BIT9	=00000200	124 (2)								
BLDCDB	0000030A-R	1340 (28)	#-1133 (21)	#-264 (6)	#-752 (18)					
BLD_RTN	00000347-R	1359 (28)	#-1343 (28)							
CDB$A_ADDR	0000C010	172 (3)	#-1396 (29)							
CDB$A_STORE	0000000C	171 (3)	#-1518 (31)	#-425 (10)						
CDB$A_TBL	00000004	169 (3)	#-1352 (29)	#-1395 (29)	#-1501 (31)					
CDB$A_VEC	00000008	170 (3)	#-1522 (31)	#-423 (10)						
CDB$BLOCK	00000000-R	179 (3)	1344 (28)	1488 (31)	1500 (31)					
CDB$L_FLAGS	00000000	168 (3)	#-1345 (28)	1355 (28)	#-1397 (29)	#-1447 (30)	1488 (31)			
			#-801 (18)	#-806 (18)	#-815 (18)	#-823 (18)	#-829 (18)			
			#-830 (18)	#-838 (18)	#-839 (18)	#-846 (18)	#-847 (18)			

				#-853 (18) #-854 (18) #-862 (18) #-863 (18) #-871 (18)
				#-880 (18) #-901 (18)
CDB$L_PTH	0000001C	176	(3)	#-761 (18)
CDB$L_ST1	00000014	173	(3)	#-1508 (31) #-1519 (31) #-569 (14) #-570 (14)
CDB$M_IE	=00000020	152	(2)	#-1345 (28) #-1397 (29) #-1447 (30)
CDB$M_MBA	=00000001	147	(2)	
CDB$M_OFFSET	=00000008	150	(2)	#-800 (18) #-805 (18) #-814 (18) #-822 (18) #-829 (18)
				#-838 (18) #-846 (18) #-854 (18) #-863 (18) #-870 (18)
				#-879 (18) #-901 (18)
CDB$M_REV	=00000010	151	(2)	#-800 (18) #-805 (18) #-814 (18) #-822 (18) #-830 (18)
				#-839 (18) #-847 (18) #-853 (18) #-862 (18) #-870 (18)
				#-879 (18)
CDB$M_UBA	=00000002	148	(2)	
CDB$M_UBI	=00000004	149	(2)	
CDB$V_IE	=00000005	146	(2)	#-1488 (31) 152 (2)
CDB$V_MBA	=00000000	141	(2)	147 (2)
CDB$V_OFFSET	=00000003	144	(2)	150 (2)
CDB$V_REV	=00000004	145	(2)	151 (2)
CDB$V_UBA	=00000001	142	(2)	148 (2)
CDB$V_UBI	=00000002	143	(2)	#-1355 (28) 149 (2)
CDB$W_RVR	0000001A	175	(3)	#-1515 (31)
CDB$W_ST2	00000018	174	(3)	#-1510 (31) 1513 (31) #-1516 (31) #-1520 (31)
CDBSIZ	=00000020	177	(3)	180 (3)
CHC$_ABORT	=00000002	128	(2)	
CHC$_CLEAR	=00000006	128	(2)	
CHC$_CLRDFT	=00000009	128	(2)	
CHC$_DSINT	=00000005	128	(2)	
CHC$_ENINT	=00000004	128	(2)	
CHC$_INITA	=00000000	128	(2)	
CHC$_INITB	=00000001	128	(2)	
CHC$_PURGE	=00000003	128	(2)	
CHC$_SELF_TEST	=0000000B	128	(2)	
CHC$_SETDFT	=00000008	128	(2)	
CHC$_STATUS	=00000007	128	(2)	
CHC$_STOP	=0000000A	128	(2)	
CHI$M_CHNINT	=00000001	128	(2)	
CHI$M_DEVINT	=00000002	128	(2)	#-1516 (31)
CHI$M_IPL	=0000007C	128	(2)	
CHI$S_IPL	=00000005	128	(2)	#-1512 (31)
CHI$V_CHNINT	=00000000	128	(2)	
CHI$V_DEVINT	=00000001	128	(2)	
CHI$V_IPL	=00000002	128	(2)	#-1512 (31)
CHM$_FORWARD	=00000000	128	(2)	128 (2)
CHM$_INVALIDATE	=00000001	128	(2)	128 (2)
CHM$_MAP	=00000002	128	(2)	128 (2)
CHM$_MFWDN	=00000002	128	(2)	
CHM$_MFWDNO	=0000000A	128	(2)	
CHM$_MFWDV	=00000003	128	(2)	
CHM$_MFWDVO	=0000000B	128	(2)	
CHM$_MREVN	=00000006	128	(2)	
CHM$_MREVNO	=0000000E	128	(2)	
CHM$_MREVV	=00000007	128	(2)	
CHM$_MREVVO	=0000000F	128	(2)	
CHM$_NFWDN	=00000000	128	(2)	
CHM$_NREVN	=00000004	128	(2)	
CHM$_OFFSET	=00000008	128	(2)	128 (2)
CHM$_REVERSE	=00000004	128	(2)	128 (2)

```
CHN$_ADR1         =0000000C      129     (2)     #-423     (10)
CHN$_ADR2         =00000010      129     (2)     #-425     (10)    #-570     (14)
CHN$_FUNC         =00000008      129     (2)     #-270     (6)
CHN$_NARGS        =00000004      129     (2)
CHN$_UNIT         =00000004      129     (2)     #-263     (6)
CHN_ABORT          0000004E-R    380     (9)       282     (6)
CHN_CALL_RTN       000000B0-R    698     (17)    #-267     (6)     #-284     (6)     #-350     (8)     #-382     (9)     #-429     (10)
                                                 #-461     (11)    #-497     (12)    #-529     (13)    #-613     (15)    #-655     (16)
CHN_CLEAR          00000084-R    527     (13)      282     (6)
CHN_CLRDFT         000000A0-R    653     (16)      282     (6)
CHN_DSINT          0000007F-R    495     (12)      282     (6)
CHN_DSP            0000000F-R    269     (6)     #-265     (6)
CHN_ENINT          00000057-R    417     (10)      282     (6)
CHN_INITA          00000035-R    314     (7)       282     (6)
CHN_INITB          00000041-R    347     (8)       282     (6)
CHN_NORMAL_RTN     000000A9-R    695     (17)    #-317     (7)     #-571     (14)
CHN_PURGE          00000076-R    459     (11)      282     (6)
CHN_SETDFT         00000097-R    611     (15)      282     (6)
CHN_STATUS         0000008D-R    568     (14)      282     (6)
CHS$M_BADBDP      =20000000      128     (2)
CHS$M_BUSIC       =00800000      128     (2)
CHS$M_BUSINIT     =00400000      128     (2)
CHS$M_BUSNXM      =08000000      128     (2)
CHS$M_BUSPDN      =01000000      128     (2)
CHS$M_CHNDPE      =00000040      128     (2)
CHS$M_CHNERR      =00000002      128     (2)       12?     (2)       156     (2)
CHS$M_CHNMPE      =00000080      128     (2)
CHS$M_CHPFOT      =00000100      128     (2)
CHS$M_DEVBUS      =00000010      128     (2)
CHS$M_DEVERR      =00000004      128     (2)       128     (2)       156     (2)
CHS$M_DEVTO       =00000020      128     (2)
CHS$M_ERRANY      =0000000F      156     (2)
CHS$M_MBAATN      =00100000      128     (2)
CHS$M_MBACPE      =00200000      128     (2)
CHS$M_MBADTB      =00040000      128     (2)
CHS$M_MBADTC      =00080000      128     (2)
CHS$M_MBAEXC      =00010000      128     (2)
CHS$M_MBAMED      =00020000      128     (2)
CHS$M_MBAWCKLWR   =02000000      128     (2)
CHS$M_MBAWCKUPR   =04000000      128     (2)
CHS$M_PGMERR      =00000008      128     (2)       128     (2)       156     (2)
CHS$M_PGMHDE      =00000800      128     (2)
CHS$M_SYSERR      =00000001      128     (2)       128     (2)       156     (2)
CHS$M_SYSMEM      =00000200      128     (2)
CHS$M_SYSSBI      =00000400      128     (2)
CHS$M_UIE         =10000000      128     (2)
CHS$V_BADBDP      =0000001D      128     (2)
CHS$V_BUSIC       =00000017      128     (2)
CHS$V_BUSINIT     =00000016      128     (2)
CHS$V_BUSNXM      =0000001B      128     (2)
CHS$V_BUSPDN      =00000018      128     (2)
CHS$V_CHNDPE      =00000006      128     (2)
CHS$V_CHNERR      =00000001      128     (2)
CHS$V_CHNMPE      =00000007      128     (2)
CHS$V_CHPFOT      =00000008      128     (2)
CHS$V_DEVBUS      =00000004      128     (2)
CHS$V_DEVERR      =00000002      128     (2)
```

```
CHS$V_DEVTO     =00000005       128     (2)
CHS$V_MBAATN    =00000014       128     (2)
CHS$V_MBACPE    =00000015       128     (2)
CHS$V_MBADTB    =00000012       128     (2)
CHS$V_MBADTC    =00000013       128     (2)
CHS$V_MBAEXC    =00000010       128     (2)
CHS$V_MBANED    =00000011       128     (2)
CHS$V_MBAWCKLWR=00000019        128     (2)
CHS$V_MBAWCKUPR=0000001A        128     (2)
CHS$V_PGMERR    =00000003       128     (2)
CHS$V_PGMHDE    =0000000B       128     (2)
CHS$V_SYSERR    =00000000       128     (2)
CHS$V_SYSMEM    =00000009       128     (2)
CHS$V_SYSSBI    =00C0000A       128     (2)
CHS$V_UIF       =0000001C       128     (2)
CLRMAP           0000021D-R     922     (18)    #-802     (18)    #-807     (18)    #-831     (18)    #-855     (18)    #-872     (18)
CLRVEC           00000379-R     1446    (30)    #-496     (12)
CNTCHK           0000024F-R     978     (18)    #-773     (18)
CNTCHK_RTN       00000276-R     990     (18)    #-987     (18)
CVT$T_MBAMAP     00000054-R     188     (3)
CVT$T_MBASR      00000050-R     185     (3)
CVT$T_UBADPR     00000058-R     191     (3)
CVT$T_UBAMAP     0000005C-R     194     (3)
DS$CVTREG        00000000-XR                            1202    (23)
DS$GPHARD        00000000-XR                            1341    (28)
DS$K_ERROR      =00000002       125     (2)
DS$K_NORMAL     =00000001       125     (2)
DS$K_SEVERE     =00000004       125     (2)
DS$K_SUBSYS     =00000066       125     (2)     104     (2)     125     (2)
DS$K_WARNING    =00000000       125     (2)
DS$PRINTB        00000000-XR                    1138    (21)    1169    (22)    1207    (23)
DS$_ARITH       =006600D0       125     (2)
DS$_ASBE        =00660118       125     (2)
DS$_BADLINK     =006600F0       125     (2)
DS$_BADTYPE     =006600E8       125     (2)
DS$_BIIC        =00660120       125     (2)
DS$_CHME        =006600A8       125     (2)
DS$_CHMK        =006600E0       125     (2)
DS$_DEVNAME     =00660108       125     (2)
DS$_ERROR       =00660002       125     (2)
DS$_FHWE        =00660068       125     (2)
DS$_FRAGBUF     =00660080       125     (2)
DS$_ICBUSY      =006600C8       125     (2)
DS$_ICERR       =006600C0       125     (2)
DS$_IHWE        =00660060       125     (2)
DS$_ILLCHAR     =00660018       125     (2)
DS$_ILLPAGCNT   =00660078       125     (2)
DS$_ILLUNIT     =00660100       125     (2)
DS$_INSFMEM     =00660050       125     (2)
DS$_IPL2HI      =006600B8       125     (2)
DS$_IVADDR      =00660040       125     (2)
DS$_IVVECT      =00660038       125     (2)
DS$_KRNLSTK     =00660090       125     (2)
DS$_LOGIC       =00660070       125     (2)
DS$_MCHK        =00660088       125     (2)
DS$_MMOFF       =00660058       125     (2)
DS$_NEEDUNIT    =006600F8       125     (2)
```

DS$_NODE	=00660128	125	(2)								
DS$_NOPCS	=00660110	125	(2)								
DS$_NORMAL	=00660001	125	(2)	#-1001	(18)	#-1088	(20)	#-1139	(21)	#-1357	(28)
				#-696	(17)	#-962	(18)	#-985	(18)	#-1405	(29)
DS$_NOSUPPORT	=006600B1	125	(2)	#-314	(7)	#-349	(8)	#-381	(9)	#-460	(11)
				#-528	(13)	#-612	(15)	#-654	(16)		
DS$_NOTDON	=00660030	125	(2)								
DS$_NOTIMP	=006600B0	125	(2)	104	(2)	#-1044	(19)	121	(2)	125	(2)
DS$_NULLSTR	=00660010	125	(2)								
DS$_OVERFLOW	=00660008	125	(2)								
DS$_POWER	=00660098	125	(2)								
DS$_PROGERR	=00660020	125	(2)	#-283	(6)	#-421	(10)	#-797	(18)	#-958	(18)
				#-988	(18)						
DS$_SEVERE	=00660004	125	(2)								
DS$_TRANSL	=006600A0	125	(2)								
DS$_TRUNCATE	=00660028	125	(2)								
DS$_UNEXPINT	=006600D8	125	(2)								
DS$_VASFULL	=00660048	125	(2)								
DS$_WARNING	=00660000	125	(2)	104	(2)	125	(2)				
DSI$CHANNEL	0000038C-R	1493	(31)								
DSI$CHANNEL_COM	0000038E-R	1495	(31)	#-1489	(31)						
DSI$CHANNEL_VEC	00000380-R	1487	(31)								
DSP$SHOWMBA	00000281-R	1043	(19)								
DSP$SHOWUBI	0000028B-R	1085	(20)								
DSX$CHANNEL	00000000-R	261	(6)								
DSX$SETMAP	000000B1-R	749	(18)								
DSX$SHOWCHAN	0000029C-R	1130	(21)								
FMT_CRLF	00000000-R	206	(4)	1138	(21)						
FMT_CVT_ARG	00000020-R	183	(3)	#-1200	(23)	#-1201	(23)	1202	(23)		
FMT_UBI_CSR	0000001F-R	208	(4)								
FMT_UBI_HDR	00000003-R	207	(4)	1169	(22)						
GENSTS	00000304-R	1266	(25)	#-1509	(31)						
HDWSTS	00000307-R	1296	(26)	#-1504	(31)						
HP$A_DEVICE	00000018			#-1353	(28)	#-1502	(31)				
HP$A_LINK	00000020			#-1347	(28)	#-1349	(28)				
HP$W_VECTOR	00000024			#-1396	(29)						
INT_DSP	000003E1-R	1524	(31)								
INVALID	0000012B-R	796	(18)	795	(18)						
INVALIDATE	00000135-R	799	(18)	795	(18)						
JMPADR	000000E9-R	223	(5)								
MBA_MAP_BIT	0000003D-R	210	(4)	189	(3)						
MBA_SR_BIT	0000003D-R	209	(4)	186	(3)						
MFWDN	00000150-R	813	(18)	795	(18)						
MFWDNO	000001B6-R	861	(18)	795	(18)						
MFWDV	0000013E-R	804	(18)	795	(18)						
MFWDVO	000001A1-R	852	(18)	795	(18)						
MREVN	00000180-R	837	(18)	795	(18)						
MREVNO	000001DA-R	878	(18)	795	(18)						
MREVV	0000016B-R	828	(18)	795	(18)						
MREVVO	000001C8-R	869	(18)	795	(18)						
NFWDN	0000015F-R	821	(18)	795	(18)						
NREVN	00000192-R	845	(18)	795	(18)						
PAR$M_[illegible]DEF	=00000010	126	(2)								
PAR$M_[illegible]	=00000008	126	(2)								
PAR$[illegible]O	=00000004	126	(2)								
PAR$M_[illegible]DEF	=00000002	126	(2)								
PAR$V_[illegible]DEF	=00000004	126	(2)								
PAR$V_ATHI	=00000003	126	(2)								

```
PAR$V_ATLO      =00000002       126     (2)
PAR$V_NODEF     =00000001       126     (2)
PAR$_BIN        =00000002       126     (2)
PAR$_DEC        =0000000A       126     (2)
PAR$_HEX        =00000010       126     (2)
PAR$_NO         =00000000       126     (2)
PAR$_OCT        =00000008       126     (2)
PAR$_YES        =00000001       126     (2)
PR$_IPL         =00000012                       #-1498  (31)
RNGCHK           0000024B-R     970     (18)    #-757   (18)
SCB_BASE         00000000-XR                      1399   (29)
SCB_UNKINT       00000000-XR                      1400   (29)
SETCNT           0000022E-R     947     (18)    #-810   (18)    #-818   (18)    #-825   (18)    #-834   (18)    #-842   (18)
                                                 #-849   (18)    #-858   (18)    #-866   (18)    #-875   (18)    #-883   (18)
SETMAP           000001E9-R     895     (18)    #-808   (18)    #-816   (18)    #-832   (18)    #-840   (18)    #-856   (18)
                                                 #-864   (18)    #-873   (18)    #-881   (18)
SETVAR           0000022D-R     937     (18)    #-809   (18)    #-817   (18)    #-824   (18)    #-833   (18)    #-841   (18)
                                                 #-848   (18)    #-857   (18)    #-865   (18)    #-874   (18)    #-882   (18)
SETVEC           00000348-R     1392    (29)    #-418   (10)
SHOWBITS         000002CE-R     1199    (23)
SHOW_UBI         000002C0-R     1168    (22)    #-1087  (20)    #-1136  (21)
SHW$K_BUFSIZ    =00000080       137     (2)      183    (3)      222    (5)
SHW$_NARGS      =00000001       132     (2)
SHW$_UNIT       =00000004       132     (2)     #-1132  (21)
SHWBUF           00000068-R     222     (5)      1203   (23)     183    (3)
SIZ...          =00000001       128     (2)      126    (2)      128    (2)
STM$_BAS        =00000010       131     (2)     #-762   (18)
STM$_CNT        =00000014       131     (2)     #-764   (18)    #-984   (18)
STM$_FUNC       =00000008       131     (2)     #-777   (18)
STM$_NARGS      =00000006       131     (2)
STM$_PHY        =0000000C       131     (2)     #-756   (18)    #-763   (18)    #-980   (18)
STM$_PTH        =00000018       131     (2)     #-761   (18)
STM$_UNIT       =00000004       131     (2)     #-751   (18)
STM_CALL_RTN     00000280-R     1003    (18)    #-754   (18)    #-759   (18)    #-771   (18)    #-775   (18)    #-798   (18)
STM_NORMAL_RTN   00000279-R     1000    (18)    #-803   (18)    #-811   (18)    #-819   (18)    #-826   (18)    #-835   (18)
                                                 #-843   (18)    #-850   (18)    #-859   (18)    #-867   (18)    #-876   (18)
                                                 #-884   (18)
UBA$S_MAP_ADDR  =00000015                        #-906   (18)
UBA$V_MAP_ADDR  =00000000                        #-905   (18)
UBA_DPR_BIT      0000003D-R     211     (4)      192    (3)
UBA_MAP_BIT      0000003E-R     213     (4)      195    (3)
UBI$L_CSR       =00000000       108     (2)
UBI$L_MAP       =00000800       109     (2)     #-907   (18)    #-924   (18)
UBI$M_MAP_BO    =02000000       113     (2)     #-903   (18)
UBI$M_MAP_VALID=80000000        111     (2)     #-900   (18)
UBI$S_MAP_BO    =00000001       114     (2)
UBI$S_MAP_DPD   =00000002       116     (2)
UBI$S_MAP_PFN   =0000000F       118     (2)
UBI$V_MAP_BO    =00000019       112     (2)      113    (2)
UBI$V_MAP_DPD   =00000015       115     (2)
UBI$V_MAP_PFN   =00000000       117     (2)
UBI$V_MAP_VALID=0000001F        110     (2)      111    (2)
UBIVEC           000000ED-R     224     (5)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CHNDEF	1	87 (2)	129 (2)
$D1_PRINT_S	1	1138 (21)	1138 (21) 1169 (22)
$DEF	1	128 (2)	
$DEFINI	1	105 (2)	105 (2) 127 (2) 130 (2)
$DS_BITDEF	2	124 (2)	124 (2)
$DS_CHCDEF	1	128 (2)	128 (2)
$DS_CHDEF	1	128 (2)	128 (2)
$DS_CHIDEF	1	128 (2)	128 (2)
$DS_CHMDEF	2	128 (2)	128 (2)
$DS_CHSDEF	3	128 (2)	128 (2)
$DS_CVTREG_G	1	1202 (23)	1202 (23)
$DS_CVTREG_L	1	183 (3)	183 (3)
$DS_DSDEF	2	125 (2)	104 (2) 125 (2)
$DS_GPHARD_S	1	1341 (28)	1341 (28)
$DS_HPODEF	2	127 (2)	127 (2)
$DS_PARDEF	1	126 (2)	126 (2)
$DS_PRINTB_S	1	1138 (21)	1138 (21) 1169 (22)
$EQU	1	128 (2)	104 (2) 124 (2) 125 (2) 126 (2) 128 (2)
$EQULS1	1	128 (2)	104 (2) 125 (2) 128 (2)
$EQULST	1	104 (2)	104 (2) 125 (2) 128 (2)
$GBLINI	2	104 (2)	104 (2) 124 (2) 125 (2) 126 (2) 128 (2)
$OFFDEF	1	129 (2)	129 (2) 131 (2) 132 (2)
$PRDEF	4	130 (2)	130 (2)
$SHWDEF	1	93 (2)	132 (2)
$STMDEF	1	90 (2)	131 (2)
$UBADEF	6	105 (2)	105 (2)
$VIELD	1	126 (2)	126 (2) 128 (2)
$VIELD1	1	128 (2)	126 (2) 128 (2)
CASE	1	271 (6)	271 (6) 778 (18)
DSBINT	1	1498 (31)	1498 (31)
MODNAM	1	200 (3)	200 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.22
Command processing	132	00:00:00.69	00:00:01.79
Pass 1	939	00:00:12.64	00:00:18.50
Symbol table sort	0	00:00:00.75	00:00:01.06
Pass 2	309	00:00:03.71	00:00:05.66
Symbol table output	28	00:00:00.21	00:00:00.77
Psect synopsis output	8	00:00:00.04	00:00:00.48
Cross-reference output	106	00:00:01.41	00:00:01.62
Assembler run totals	1558	00:00:19.58	00:00:30.11

The working set limit was 1000 pages.
70128 bytes (137 pages) of virtual memory were used to buffer the intermediate code.

There were 30 pages of symbol table space allocated to hold 525 non-local and 18 local symbols.
1530 source lines were read in Pass 1, producing 0 object records in Pass 2.
95 pages of virtual memory were used to define 25 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	15
DRB1:[DS.WORK]DS.MLB;218	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	26

619 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CHAN730/UPDA=(CHAN730.UPD,CHAN730.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SY
```

Table of contents

(1) 54 DECLARATIONS
(2) 85 CHANGE MODE TO KERNEL ROUTINE
(3) 155 ASTEXIT SYSTEM SERVICE
(3) 181 Local intermediate dispatches

```
0000      1          .TITLE  CHMK    *** CHMK Change mode to kernel handler
0000      2          .IDENT  /07-07/
0000      3          .LIST   MEB
0000      4          .NLIST  CND
0000      5
0000      6 ;
0000      7 ; COPYRIGHT (c) 1977, 1981
0000      8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      9 ;
0000     10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     15 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     16 ; REMAIN IN DEC.
0000     17 ;
0000     18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     20 ; CORPORATION.
0000     21 ;
0000     22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     24
0000     25 ;++
0000     26 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     27 ;
0000     28 ; ABSTRACT:
0000     29 ;
0000     30 ; ENVIRONMENT:
0000     31 ;
0000     32 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000     33 ;
0000     34 ; MODIFIED BY:
0000     35 ;                TOM SOUTTER     29-MAR-78       VERSION 02 (ESSAA-4.01)
0000     36 ;       01       ADDED CHMK HANDLER FOR ASTEXIT
0000     37 ;                D Butenhof      16-jan-80
0000     38 ;       02       Add dispatch for SGIPR (Store/Get Int. Proc. Reg.)
0000     39 ;
0000     40 ;                Roger Riggs,    12-Mar-1980
0000     41 ;       03       Removed CMK$MMON, CMK$MMOFF, added CMK$MMENABLE
0000     42 ;                Dave Butenhof, 23-feb-1981, version 6.3
0000     43 ;       04       Delete SEP psect, fix trunc. errors
0000     44 ;       05       Make dispatch offsets .SIGNED_WORD to report signed
0000     45 ;                truncation errors
0000     46 ;                - Dave Butenhof, 08-Jun-1981, version 6.4
0000     47 ;       06       Add dispatch for CHK$_SETPRT
0000     48 ;
0000     49 ;       07       - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     50 ;                Added .LIBRARY statements for $DS and $DIAG.
0000     51 ;                Fixed a truncation error.
0000     52 ;--
```

```
                                0000     54          .SBTTL  DECLARATIONS
                                0000     55 ;
                                0000     56 ; INCLUDE FILES:
                                0000     57 ;
                                0000     58          .LIBRARY        /$DS/           ;                                       [07]
                                0000     59          .LIBRARY        /$DIAG/         ;                                       [07]
                                0000     60
                                0000     61 ;
                                0000     62 ; MACROS:
                                0000     63 ;
                                0000     64
                                0000     65 ;
                                0000     66 ; EQUATED SYMBOLS:
                                0000     67 ;
                                C000     68          CMKDEF
                                0000     69          DSFDEF
                                0000     73          $PSLDEF
                                0000     74          $PCBDEF
                                0000     78
                                0000     79 ;
                                0000     80 ; OWN STORAGE:
                                0000     81 ;
                           00000000     82          .PSECT  DATA, NOEXE, SHR, NOWRT, LONG
                                0000     83          MODNAM  CHMK
                    4B 4D 48 43 00' 0000         $MODULE:        .ASCIC  \CHMK\
                                04  0000
```

```
                        85          .SBTTL  CHANGE MODE TO KERNEL ROUTINE
                        86 ;++
                        87 ; FUNCTIONAL DESCRIPTION:
                        88 ;
                        89 ;       THIS ROUTINE DISPATCHES TO THE PROPER PRIVILEGED ROUTINE.
                        90 ;
                        91 ; CALLING SEQUENCE:
                        92 ;
                        93 ;       NONE
                        94 ;
                        95 ; INPUT PARAMETERS:
                        96 ;
                        97 ;       NONE
                        98 ;
                        99 ; IMPLICIT INPUTS:
                       100 ;
                       101 ;       NONE
                       102 ;
                       103 ; OUTPUT PARAMETERS:
                       104 ;
                       105 ;       NONE
                       106 ;
                       107 ; IMPLICIT OUTPUTS:
                       108 ;
                       109 ;       NONE
                       110 ;
                       111 ; COMPLETION CODES:
                       112 ;
                       113 ;       NONE
                       114 ;
                       115 ; SIDE EFFECTS:
                       116 ;
                       117 ;       NONE
                       118 ;
                       119 ;--
```

```
                        00000000   121          .PSECT  WORK, NOEXE, NOSHR, WRT, LONG
                              0000 122
                        00000000   0000 123 CMKCODE:    .LONG   0               ; SAVED CODE FOR USER VECTOR.
                              0004 124
                              0004 125
                        00000000   126          .PSECT  CODE, EXE, SHR, NOWRT, LONG
                              0000 127          .ALIGN  LONG,0
                              0000 128 EXE_CMODKRNL::
00000000'EF    6E    D0   0000 129          MOVL    (SP), CMKCODE
    0D   00    8E    CF   0007 130          CASEL   (SP)+, #0, #CMK$_LAST-1 ; DISPATCH.
                       0049' 000B 131 10$:          .SIGNED_WORD    CMK$ASTEXIT-10$
                       FFF5' 000D 132               .SIGNED_WORD    CMK$RCONSOLE-10$
                       FFF5' 000F 133               .SIGNED_WORD    CMK$TCONSOLE-10$
                       0066' 0011 134               .SIGNED_WORD    MMENABLE-10$
                       001C' 0013 135               .SIGNED_WORD    20$-10$         ; Unused change mode code
                       FFF5' 0015 136               .SIGNED_WORD    CMK$REFRESH-10$
                       001C' 0017 137               .SIGNED_WORD    20$-10$
                       FFF5' 0019 138               .SIGNED_WORD    CMK$HIBER-10$
                       006C' 001B 139               .SIGNED_WORD    INITSCB-10$
                       0072' 001D 140               .SIGNED_WORD    SETIPL-10$
                       FFF5' 001F 141               .SIGNED_WORD    CMK$SGIPR-10$
                       FFF5' 0021 142               .SIGNED_WORD    CMK$CANTIM-10$  ; Cancel timer kernel mode rtn
                       FFF5' 0023 143               .SIGNED_WORD    CMK$SETIMR-10$  ; Setimr kernel mode routine
                       0078' 0025 144               .SIGNED_WORD    SETPRT-10$      ; Set page protection
                             0027 145
     00000000'EF       D5   0027 146 20$:     TSTL    L^DS$GA_CHMKVEC         ; Check for soft vector
                 19    12   002D 147          BNEQ    30$
50   00000000'EF       D0   002F 148          MOVL    CMKCODE,R0              ; Get code so ERRSUP prints it
                             0036 149          ERRSUP_S
     00000000'EF       DF   0036                      PUSHAL  $MODULE
                 00    DD   003C                      PUSHL   #0
                 01    DD   003E                      PUSHL   #$ER
00000000'EF      03    FB   0040                      CALLS   #3, DS_ERRSUP
                       02   0047 150          REI
                            0048 151
     00000000'EF       DD   0048 152 30$:     PUSHL   CMKCODE                 ; RESTORE CODE ON STACK.
     00000000'FF       17   004E 153          JMP     @L^DS$GA_CHMKVEC        ; Go through user vector
```

```
                                0054   155          .SBTTL  ASTEXIT SYSTEM SERVICE
                                0054   156 ;+
                                0054   157 ; CMK$ASTEXIT - SERVICE TO EXIT AN ACTIVE AST AND ALLOW PENDING ASTS TO
                                0054   158 ;                BE DELIVERED.
                                0054   159 ;
                                0054   160 ; THIS SYSTEM SERVICE IS INVOKED WITH A ''CMK ASTEXIT'' NOT CONTAINED IN
                                0054   161 ; A STANDARD SYSTEM SERVICE VECTOR TO AVOID CLUTTERING THE STACK WITH AN
                                0054   162 ; ADDITIONAL CALL FRAME DURING AST EXIT PROCESSING.
                                0054   163 ;
                                0054   164 ; INPUTS:
                                0054   165 ;       NONE
                                0054   166 ;
                                0054   167 ; OUTPUTS:
                                0054   168 ;       PCB$B_ASTACT IS CLEARED FOR THE ISSUING MODE
                                C054   169 ;
                                0054   170 ;-
                                0054   171
                                0054   172 CMK$ASTEXIT::                                ;EXIT ACTIVE AST
 50   04 AE   02   16   EF      0054   173          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,4(SP),R0 ;GET PREVIOUS MODE
                14   BB   005A   174          PUSHR   #^M<R2,R4>                  ;SAVE R2,R4
      54   00000000'EF   DE   005C   175          MOVAL   DS$AX_SOFTPCB,R4            ;GET PCB CURRENT PCB ADDRESS
         00 0C A4   50   E7   0063   176          BBCCI   R0,PCB$B_ASTACT(R4),10$ ;CLEAR AST ACTIVE BIT FOR MODE
           00000000'EF   16   0068   177 10$:     JSB     SCH$NEWLVL                  ;COMPUTE NEW AST LEVEL SETTING          [07]
                14   BA   006E   178          POPR    #^M<R2,R4>                  ;RESTORE REGISTERS<R2,R4>
                     02   0070   179          REI                                 ;AND EXIT
```

```
                                0071   181          .SBTTL  Local intermediate dispatches
                                0071   182 ;+
                                0071   183 ; Since CASE offsets have to be word relative, to fix truncation errors,
                                0071   184 ; dispatch to local JMP L^ way-stations.
                                0071   185 ;-
                                0071   186
                                0071   187 MMENABLE:
00000000'EF    17               0071   188          JMP     L^CMK$MMENABLE          ; Dispatch memory management on
                                0077   189
                                0077   190 INITSCB:
00000000'EF    17               0077   191          JMP     L^CMK$INITSCB           ; Dispatch initialize SCB
                                007D   192
00000000'EF    17               007D   193 SETIPL: JMP     L^CMK$SETIPL            ; Dispatch set IPL
                                0083   194
                                C083   195 SETPRT:
00000000'EF    17               0083   196          JMP     L^CMK$SETPRT            ; Dispatch the trap
                                0089   197
                                0089   198          .END
```

```
$ER                     =  00000002     D
$MODULE                    00000000 R   D   02
CMK$ASTEXIT                00000054 RG  D   04
CMK$CANTIM                 ********     X   04
CMK$HIBER                  ********     X   04
CMK$INITSCB                ********     X   04
CMK$MMENABLE               ********     X   04
CMK$RCONSOLE               ********     X   04
CMK$REFRESH                ********     X   04
CMK$SETIMR                 ********     X   04
CMK$SETIPL                 ********     X   04
CMK$SETPRT                 ********     X   04
CMK$SGIPR                  ********     X   04
CMK$TCONSOLE               ********     X   04
CMK$_                   =  00000004     D
CMK$_ASTEXIT            =  00000000     D
CMK$_CANTIM             =  00000008     D
CMK$_HIBER              =  00000007     D
CMK$_INITSCB            =  00000008     D
CMK$_LAST               =  0000000E     D
CMK$_MMENABLE           =  00000003     D
CMK$_RCONSOLE           =  00000001     D
CMK$_REFRESH            =  00000005     D
CMK$_SCHDWK             =  00000006     D
CMK$_SETIMR             =  0000000C     D
CMK$_SETIPL             =  00000009     D
CMK$_SETPRT             =  0000000D     D
CMK$_SGIPR              =  0000000A     D
CMK$_TCONSOLE           =  00000002     D
CMKCODE                    00000000 R   D   03
DS$AX_SOFTPCB              ********     X   04
DS$GA_CHMKVEC              ********     X   04
DS$M_ABRTFLG            =  00000040     D
DS$M_BADTIME            =  00100000     D
DS$M_BATCH              =  00400000     D
DS$M_BRKCLR             =  00001000     D
DS$M_BRKPT              =  00000800     D
DS$M_CHARFLG            =  00000100     D
DS$M_CMDFLG             =  00000080     D
DS$M_CTRLC              =  00000001     D
DS$M_CTRLO              =  00010000     D
DS$M_DEVFLG             =  00000200     D
DS$M_DISABLCC           =  01000000     D
DS$M_DONFLG             =  00002000     D
DS$M_ERRFLG             =  00000010     D
DS$M_EXCEPT             =  00080000     D
DS$M_EXETST             =  00040000     D
DS$M_HLTFLG             =  00000008     D
DS$M_LODFLG             =  00000002     D
DS$M_MEMMGT             =  00008000     D
DS$M_OUTPUT             =  00800000     D
DS$M_RUBFLG             =  00000020     D
DS$M_SCRIPT             =  00200000     D
DS$M_SETIMR             =  02000000     D
DS$M_STRFLG             =  00000004     D
DS$M_SUBT               =  00004000     D
DS$M_SYSFLG             =  00000400     D
DS$M_TIMRON             =  00020000     D
DS$V_ABRTFLG            =  00000006     D
DS$V_BADTIME            =  00000014     D
DS$V_BATCH              =  00000016     D
DS$V_BRKCLR             =  0000000C     D
DS$V_BRKPT              =  0000000B     D
DS$V_CHARFLG            =  00000008     D
DS$V_CMDFLG             =  00000007     D
DS$V_CTRLC              =  00000000     D
DS$V_CTRLO              =  00000010     D
DS$V_DEVFLG             =  00000009     D
DS$V_DISABLCC           =  00000018     D
DS$V_DONFLG             =  0000000D     D
DS$V_ERRFLG             =  00000004     D
DS$V_EXCEPT             =  00000013     D
DS$V_EXETST             =  00000012     D
DS$V_HLTFLG             =  00000003     D
DS$V_LODFLG             =  00000001     D
DS$V_MEMMGT             =  0000000F     D
DS$V_OUTPUT             =  00000017     D
DS$V_RUBFLG             =  00000005     D
DS$V_SCRIPT             =  00000015     D
DS$V_SETIMR             =  00000019     D
DS$V_STRFLG             =  00000002     D
DS$V_SUBT               =  0000000E     D
DS$V_SYSFLG             =  0000000A     D
DS$V_TIMRON             =  00000011     D
DS_ERRSUP                  ********     X   04
EXE_CMODKRNL               00000000 RG  D   04
INITSCB                    00000077 R   D   04
MMENABLE                   00000071 R   D   04
PCB$B_ASTACT               0000000C     D
PCB$B_ASTEN                0000000D     D
PCB$B_PRI                  0000000B     D
PCB$B_PRIB                 0000002F     D
PCB$B_TYPE                 0000000A     D
PCB$B_WEFC                 0000002E     D
PCB$C_LENGTH               0000008C     D
PCB$K_LENGTH               0000008C     D
PCB$L_ARB                  00000084     D
PCB$L_ASTQBL               00000014     D
PCB$L_ASTQFL               00000010     D
PCB$L_EFC2P                00000058     D
PCB$L_EFC3P                0000005C     D
PCB$L_EFCS                 00000050     D
PCB$L_EFCU                 00000054     D
PCB$L_EFWM                 0000004C     D
PCB$L_JIB                  00000078     D
PCB$L_OWNER                0000001C     D
PCB$L_PHD                  00000064     D
PCB$L_PHYPCB               00000018     D
PCB$L_PID                  00000060     D
PCB$L_PQB                  0000004C     D
PCB$L_SQBL                 00000004     D
PCB$L_SQFL                 00000000     D
PCB$L_STS                  00000024     D
PCB$L_UIC                  00000088     D
PCB$L_WSSWP                00000020     D
PCB$L_WTIME                00000028     D
PCB$Q_PRIV                 0000007C     D
PCB$T_LNAME                00000068     D
PCB$T_TERMINAL             00000044     D
PCB$W_APTCNT               00000030     D
PCB$W_ASTCNT               00000038     D
PCB$W_BIOCNT               0000003A     D
PCB$W_BIOLM                0000003C     D
PCB$W_DIOCNT               0000003E     D
PCB$W_DIOLM                00000040     D
PCB$W_GPGCNT               00000034     D
PCB$W_GRP                  0000008A     D
PCB$W_MEM                  00000088     D
PCB$W_MTXCNT               0000000E     D
PCB$W_PPGCNT               00000036     D
PCB$W_PRCCNT               00000042     D
PCB$W_SIZE                 00000008     D
PCB$W_STATE                0000002C     D
PCB$W_TMBU                 00000032     D
PSL$S_PRVMOD            =  00000002     D
PSL$V_PRVMOD            =  00000016     D
SCH$NEWLVL                 ********     X   04
SETIPL                     0000007D R   D   04
SETPRT                     00000083 R   D   04
SIZ...                  =  00000001     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (140.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000005 (5.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	00000004 (4.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000089 (137.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...					
$ER	=00000002	149	(3)	#-149	(3)				
$MODULE	00000000-R	83	(1)	149	(3)				
BIT...	=00000018	69	(1)	68	(1)	69	(1)		
CMK$ASTEXIT	00000054-R	172	(3)	131	(3)				
CMK$CANTIM	00000000-XR			142	(3)				
CMK$HIBER	00000000-XR			138	(3)				
CMK$INITSCB	00000000-XR			191	(3)				
CMK$MMENABLE	00000000-XR			188	(3)				
CMK$RCONSOLE	00000000-XR			132	(3)				
CMK$REFRESH	00000000-XR			136	(3)				
CMK$SETIMR	00000000-XR			143	(3)				
CMK$SETIPL	00000000-XR			193	(3)				
CMK$SETPRT	00000000-XR			196	(3)				
CMK$SGIPR	00000000-XR			141	(3)				
CMK$TCONSOLE	00000000-XR			133	(3)				
CMK$_	=00000004	68	(1)						
CMK$_ASTEXIT	=00000000	68	(1)						
CMK$_CANTIM	=0000000B	68	(1)						
CMK$_HIBER	=00000007	68	(1)						
CMK$_INITSCB	=00000008	68	(1)						
CMK$_LAST	=0000000E	68	(1)	#-130	(3)				
CMK$_MMENABLE	=00000003	68	(1)						
CMK$_RCONSOLE	=00000001	68	(1)						
CMK$_REFRESH	=00000005	68	(1)						
CMK$_SCHDWK	=00000006	68	(1)						
CMK$_SETIMR	=0000000C	68	(1)						
CMK$_SETIPL	=00000009	68	(1)						
CMK$_SETPRT	=0000000D	68	(1)						
CMK$_SGIPR	=0000000A	68	(1)						
CMK$_TCONSOLE	=00000002	68	(1)						
CMKCODE	00000000-R	123	(3)	#-129	(3)	#-148	(3)	#-152	(3)
DS$AX_SOFTPCB	00000000-XR			175	(3)				
DS$GA_CHMKVEC	00000000-XR			#-146	(3)	153	(3)		
DS$M_ABRTFLG	=00000040	69	(1)						
DS$M_BADTIME	=00100000	69	(1)						
DS$M_BATCH	=00400000	69	(1)						
DS$M_BRKCLR	=00001000	69	(1)						
DS$M_BRKPT	=00000800	69	(1)						
DS$M_CHARFLG	=00000100	69	(1)						
DS$M_CMDFLG	=00000080	69	(1)						
DS$M_CTRLC	=00000001	69	(1)						
DS$M_CTRLO	=00010000	69	(1)						
DS$M_DEVFLG	=00000200	69	(1)						
DS$M_DISABLCC	=01000000	69	(1)						
DS$M_DONFLG	=00002000	69	(1)						
DS$M_ERRFLG	=00000010	69	(1)						
DS$M_EXCEPT	=00080000	69	(1)						
DS$M_EXETST	=00040000	69	(1)						
DS$M_HLTFLG	=00000008	69	(1)						
DS$M_LODFLG	=00000002	69	(1)						

```
DS$M_MEMMGT     =00008000      69      (1)
DS$M_OUTPUT     =00800000      69      (1)
DS$M_RUBFLG     =00000020      69      (1)
DS$M_SCRIPT     =00200000      69      (1)
DS$M_SETIMR     =02000000      69      (1)
DS$M_STRFLG     =00000004      69      (1)
DS$M_SUBT       =00004000      69      (1)
DS$M_SYSFLG     =00000400      69      (1)
DS$M_TIMRON     =00020000      69      (1)
DS$V_ABRTFLG    =00000006      69      (1)
DS$V_BADTIME    =00000014      69      (1)
DS$V_BATCH      =00000016      69      (1)
DS$V_BRKCLR     =0000000C      69      (1)
DS$V_BRKPT      =0000000B      69      (1)
DS$V_CHARFLG    =00000008      69      (1)
DS$V_CMDFLG     =00000007      69      (1)
DS$V_CTRLC      =00000000      69      (1)
DS$V_CTRLO      =00000010      69      (1)
DS$V_DEVFLG     =00000009      69      (1)
DS$V_DISABLCC   =00000018      69      (1)
DS$V_DONFLG     =0000000D      69      (1)
DS$V_ERRFLG     =00000004      69      (1)
DS$V_EXCEPT     =00000013      69      (1)
DS$V_EXETST     =00000012      69      (1)
DS$V_HLTFLG     =00000003      69      (1)
DS$V_LODFLG     =00000001      69      (1)
DS$V_MEMMGT     =0000000F      69      (1)
DS$V_OUTPUT     =00000017      69      (1)
DS$V_RUBFLG     =00000005      69      (1)
DS$V_SCRIPT     =00000015      69      (1)
DS$V_SETIMR     =00000019      69      (1)
DS$V_STRFLG     =00000002      69      (1)
DS$V_SUBT       =0000000E      69      (1)
DS$V_SYSFLG     =0000000A      69      (1)
DS$V_TIMRON     =00000011      69      (1)
DS_ERRSUF        00000000-XR                    149     (3)
EXE_CMODKRNL     00000000-R    128     (3)
INITSCB          00000077-R    190     (3)      139     (3)
MMENABLE         00000071-R    187     (3)      134     (3)
PCB$B_ASTACT     0000000C                       176     (3)
PSL$S_PRVMOD    =00000002                     #-173     (3)
PSL$V_IS        =0000001A      68      (1)
PSL$V_PRVMOD    =00000016                     #-173     (3)
                                               177     (3)
SCH$NEWLVL       00000000-XR                    140     (3)
SETIPL           0000007D-R    193     (3)      144     (3)
SETPRT           00000083-R    195     (3)       69     (1)
SIZ...          =00000001      69      (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...			
$DEF	1	69 (1)				
$DEFINI	1	73 (1)	73 (1)		74 (1)	
$EQU	1	69 (1)	68 (1)			
$EQULS1	1	68 (1)	68 (1)			
$EQULST	1	68 (1)	68 (1)			
$GBLINI	2	68 (1)	68 (1)		69 (1)	
$PCBDEF	4	74 (1)	74 (1)			
$PSLDEF	2	73 (1)	73 (1)			
$PUSHADR	1	149 (3)	149 (3)			
$VIFLD	1	69 (1)	69 (1)			
$VIELD1	1	69 (1)	69 (1)			
CMKDEF	1	68 (1)	68 (1)			
DSFDEF	3	69 (1)	69 (1)			
ERRSUP_S	1	149 (3)	149 (3)			
MODNAM	1	83 (1)	83 (1)			

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.12	00:00:00.24
Command processing	148	00:00:00.74	00:00:01.68
Pass 1	449	00:00:04.77	00:00:06.55
Symbol table sort	0	00:00:00.27	00:00:00.35
Pass 2	87	00:00:00.85	00:00:01.04
Symbol table output	13	00:00:00.10	00:00:00.10
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	33	00:00:00.37	00:00:00.41
Assembler run totals	772	00:00:07.26	00:00:10.40

The working set limit was 1000 pages.
35233 bytes (69 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 202 non-local and 4 local symbols.
198 source lines were read in Pass 1, producing 0 object records in Pass 2.
33 pages of virtual memory were used to define 15 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	1
DRB1:[DS.WORK]DS.MLB;218	4
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	12

308 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CHMK/UPDA=(CHMK.UPD,CHMK.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DI

Table of contents

(1)	339	Libraries, Macros, Equated Symbols
(1)	374	Work Psect Storage
(1)	389	Data Psect Storage
(1)	434	Commands A-C
(1)	495	Directory, Deattach, Deselect, Deposit Commands
(1)	561	Commands Exit, Examine
(1)	590	Help, InitPCS, Load, Next, Run
(1)	632	Commands Set, Select, Show, Start, Summary, Xdelta, @
(1)	685	End of Line Processing
(4)	707	Illegal Command Processing
(4)	754	Incomplete Command Processing
(4)	766	Bad List Processing
(4)	796	Bad Qualifier Processing
(4)	825	Get Generic Device Name
(4)	854	Deattach Processing
(4)	882	Deselect and Select Processing
(4)	904	Get a File Specification
(4)	987	Start and Run Processing
(4)	1107	Start and Run Qualifiers
(4)	1194	Show Parameters and Qualifiers
(5)	1518	Clear Parameters and Qualifiers
(5)	1708	Set Parameters and Qualifiers
(5)	2011	Examine Qualifiers and Parameters
(6)	2173	Deposit Parameters and Qualifiers
(6)	2335	Set Default Parameters
(6)	2410	DS$Cli Routine - Command Line Interpreter
(6)	2574	CLI Action Routines.
(6)	2743	Null and Context-Saving Action Routines
(7)	2780	Success, Failure, Notnuf, Enuf Action Routines
(7)	2811	Commands A-C Action Routines
(7)	2856	Commands D-E Action Routines
(7)	2929	Commands H-R Action Routines
(7)	3000	Commands S-Z Action Routines
(7)	3056	CRD command
(7)	3147	Filespec, Optional, Required Action Routines
(7)	3180	Start and Run Qualifier Action Routines
(7)	3293	Event and Flags Action Routines
(7)	3346	Base, Address, Break Action Routines
(7)	3390	All, Default Action Routines
(7)	3430	Show Calls Routine ; [70]
(7)	3442	CRDTrace Action Routines ; [65]
(7)	3486	Bell, Binary, Halt, !Ex Action Routines
(10)	3547	Loop, Oper, Prompt Quick Action Routines
(10)	3581	Search, Trace, Verify Action Routines
(10)	3607	Set/Show Page, Width Action Routine
(10)	3643	Set/Show QA Action Routines
(10)	3691	QADef Action Routine
(10)	3717	Efn, Next, Ascii Action Routines
(10)	3754	Register Action Routines
(10)	3798	Backup, Advance, Data Action Routines
(10)	3825	Long, Word, Byte, Hex, Dec, Oct Action Routines
(10)	3875	IF Action Routines
(10)	3968	Xdelta Action Routines
(10)	3990	Device, Brief, Adaptor Action Routines
(10)	4043	Show Select, Do_Cmd, Set Load, Show Load Action Routines
(10)	4080	MM On, Off, and Show Action Routines
(10)	4143	Show Status, Show Support

```
0000      1          .Title  CLI     DIAGNOSTIC SUPERVISOR COMMAND LINE INTERPRETER
0000      2          .Ident  /07-80/
0000      3          .NoShow Conditionals
0000      4
0000      5  ;++
0000      6  ; Copyright (c) 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984
0000      7  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      8  ;
0000      9  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     10  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     11  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     12  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     13  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     14  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     15  ; REMAIN IN DEC.
0000     16  ;
0000     17  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     18  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     19  ; CORPORATION.
0000     20  ;
0000     21  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     22  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     23  ;
0000     24  ;--
0000     25  ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     26  ;
0000     27  ; ABSTRACT:
0000     28  ;
0000     29  ; ENVIRONMENT:
0000     30  ;
0000     31  ; AUTHOR:        KEN CHAPMAN     31-OCT-77       VERSION 01.
0000     32  ;
0000     33  ; MODIFIED BY:
0000     34  ;                 NICK HOWGATE    06-NOV-77       VERSION 02
0000     35  ;       01        APT INTERFACE
0000     36  ;
0000     37  ;                 TOM SOUTTER     17-NOV-77       VERSION 03.
0000     38  ;       02        DSPR #7 - FIX FOR SHOW EVENT FLAGS.
0000     39  ;       03        MAKE 'ILLEGAL COMMAND' MESSAGE GLOBAL.
0000     40  ;
0000     41  ;                 KEN CHAPMAN     05-DEC-77       VERSION 04 (ESSAA-3.05)
0000     42  ;       04        DSPR #45 - REMOVE HALTI.
0000     43  ;
0000     44  ;                 KEN CHAPMAN     12-JAN-78       VERSION 05 (ESSAA-3.06)
0000     45  ;       05        DSPR #77 - FIXED VERSION NUMBER BRANCH OF TREE.
0000     46  ;       06        DSPR #100 - FIXED HALT BRANCH OF TREE.
0000     47  ;
0000     48  ;                 KEN CHAPMAN     16-FEB-78       VERSION 06 (ESSAA-3.07)
0000     49  ;       07        DSPR #104 - FIXED CLEAR BREAK BRANCH OF TREE.
0000     50  ;       08        ADDED 'COMPAT' FLAG.
0000     51  ;       09        CODE COMPRESSION USING CLI$_STRING MECHANISM.
```

```
0000     53 ;                KEN CHAPMAN     28-FEB-78       VERSION 07 (ESSAA-4.00)
0000     54 ;        10      SET COMMAND MODE FLAG ON ENTERING CLI.
0000     55 ;
0000     56 ;                NICK HOWGATE    29-MAR-78       VERSION 08 (ESSAA-4.01)
0000     57 ;        11      FIXED 'CONTINUE' IN APT MODE
0000     58 ;        12      ADDED PROPER SPELLING OF ''HEXADECIMAL''.
0000     59 ;        13      CHANGED 'VRRUN' & 'VRLOAD' TO 'DSV$RUN' & 'DSV$LOAD'
0000     60 ;                KEN CHAPMAN     06-APR-78
0000     61 ;        14      MADE 'COMPAT' COMMANDS CONDITIONAL ON 'VM11' DEFINED.
0000     62 ;                NICK HOWGATE    09-JUN-78       VERSION 09 (ESSAA-4.03).
0000     63 ;        15      FIX APT ^C PROBLEM
0000     64 ;                Roger Riggs     17-JUL-78       VERSION 10 (ESSAA-5.00).
0000     65 ;        16      Fix to let SUPERLINE do the prompt, clean up APT
0000     66 ;                stuff in area of prompt.
0000     67 ;                Roger Riggs     21-SEP-78
0000     68 ;        17      Insert support for new device selection commands.
0000     69 ;                Roger Riggs     04-FEB-79
0000     70 ;        18      Insert ''SET LOAD <filespec>'' command for default load device
0000     71 ;        22      Add SHOW LOAD command
0000     72 ;        23      Fix SET LOAD, SET LOOP, SET LOCK interaction
0000     73 ;                fix SELECT and DESELECT
0000     74 ;                Fix SE AMBIGIOUS error
0000     75 ;        24      Raise ASTLVL in command mode to prevent AST delivery
0000     76 ;                in command mode
0000     77 ;                Roger Riggs     4-Sept-1979
0000     78 ;        25      Made command tree and command action routine global for APT
0000     79 ;                Roger Riggs     6-Nov-1979
0000     80 ;        26      Changed version number on filespec to be decimal instead
0000     81 ;                of octal.
0000     82 ;                Changed MTPR x,PR$_IPL to call set so it's done in KERNEL mode
0000     83 ;        27      MARION BAGGETT  14-DEC-79
0000     84 ;                ALLOW SETTING OF MEMORY MANAGEMENT ON OR OFF
0000     85 ;                ALLOW EXAMINE AND DEPOSIT OF PROCESSOR REGISTERS
0000     86 ;
0000     87 ;        28      David Butenhof  09-Jan-80
0000     88 ;                Fix exam/deposit of processor registers
0000     89 ;
0000     90 ;        29      David Butenhof  06-feb-80
0000     91 ;                Be sure all possible AST levels are disabled to prevent
0000     92 ;                interrupt-driven diagnostic from continuing after ^C or
0000     93 ;                breakpoint.
0000     94 ;        30      David Butenhof  15-feb-80
0000     95 ;                Add SEARCH operator flag
0000     96 ;                Roger Riggs, 17-Jan-1980, Version 5.2
0000     97 ;        31      Corrected handling of command mode for APT, made sure
0000     98 ;                it typed ''DS> '' after ^C and processing of any command.
0000     99 ;        32      Fixed saved IPL to save all 5 bits of IPL instead of just 4.
0000    100 ;        33      Remove disable of ASTs in CLI so that WAITMS works and
0000    101 ;                ^C ASTs are delivered.
0000    102 ;                Dave Butenhof   30-jun-1980, version 5.5
0000    103 ;        34      Add HELP command in PARSE tree, and ACT_HELP action routine
```

```
0000   105 ;                Dave Butenhof, 02-oct-1980, Version 6.1
0000   106 ;        35      Add DIRECTORY command
0000   107 ;        36      Alter filespec parsing to recognise [000000] form as well as
0000   108 ;                [0,0] for UIC.
0000   109 ;                Dave Butenhof, 13-feb-1981, version 6.3
0000   110 ;        37      Change /SECTION:xxx so xxx can be symbol (inc. $ and _).
0000   111 ;        38      Add SET WIDTH command, and alter filespec parse to allow
0000   112 ;                subdirectories in user mode.
```

```
0000   114 ;
0000   115 ;       39      - dave butenhof, 06-Oct-1981, version 6.5
0000   116 ;               Implement SHOW STATUS command
0000   117 ;
0000   118 ;       40      - Jack Stansbury, 21-Oct-1981, Version 6.5
0000   119 ;               Added calls to QA$Main for the QA enhancement.
0000   120 ;               Also added .LIBRARY statements for $DIAG, $DS, and LIB.
0000   121 ;
0000   122 ;       41      - Jack Stansbury, 22-Oct-1981, Version 6.5
0000   123 ;               Fixed truncation errors.
0000   124 ;
0000   125 ;       42      - Jack Stansbury, 25-Oct-1981, Version 6.5
0000   126 ;               Many changes for the /QA qualifier on the
0000   127 ;               RUN and START commands, additional SET and SHOW
0000   128 ;               changes for the QA flags.
```

```
0000    130 ;       43      - Jack Stansbury, 3-Nov-1981, Version 6.5
0000    131 ;               Major revisions to the parse tree. Renumbered all the labels,
0000    132 ;               moved things around, ... Also incorporated checks so that a
0000    133 ;               SET FLAG WIDTH 130 is not allowed (as it was before). Also
0000    134 ;               changed the error messages to mixed case. Also took out T_CRLF,
0000    135 ;               because it was not used. Most of these changes are not marked
0000    136 ;               since so much was done. Also made use of the new CLI$K_SLASH,
0000    137 ;               CLI$K_VALUE, and CLI$K_COMMA character values in calls to
0000    138 ;               $DS_CLI. These allow "/", ":" or "=", and ",", respectively,
0000    139 ;               with optional spaces on either side of them. They were added
0000    140 ;               to PARSE.MAR by Dave Butenhof. Also added /ADAPTER qualifier
0000    141 ;               to the SHOW DEVICE, SELECT, and DESELECT commands. This qualifier
0000    142 ;               requires an adaptor name as its value. Also added SHOW DEFAULT
0000    143 ;               and SHOW BASE commands. Also added DCL-like printout of portion
0000    144 ;               of command line in error. That is,
0000    145 ;                       ?? error message
0000    146 ;                          \portion in error\
0000    147 ;
0000    148 ;       44      - Marion Baggett, Jack Stansbury, 05-Oct-1981, Version 6.5
0000    149 ;               Enhanced the SHOW DEVICE AND SHOW SELECTED command to allow
0000    150 ;               the qualifier and print only the device type and name.
0000    151 ;               Added a new command SHOW SUPPORT to show the devices a diagnostic
0000    152 ;               will support.
```

```
0000   154 ;        45      - Jack Stansbury, 11-Nov-1981, Version 6.5
0000   155 ;                Changed the DS$PARSE call to DSX$SUPERPARSE. SUPERPARSE does not
0000   156 ;                return when it encounters EOL. Rather, it continues following the
0000   157 ;                parse tree until it encounters CLI$K_EXIT. This takes care of the
0000   158 ;                problem with SET BASE 0, where DS$PARSE saw EOL, and returned to
0000   159 ;                here with the NOTNUF flag still set. Thus, it was not continuing
0000   160 ;                on in the parse tree, where it would have executed the ENUF action
0000   161 ;                routine, and then gone to EOL in the parse tree. DSX$SUPERPARSE
0000   162 ;                does continue, and thus will execute ENUF, and then go to EOL.
0000   163 ;                Also: changed the error handling and reporting. Established
0000   164 ;                different action routines for Illegal command versus Incomplete
0000   165 ;                commands. Checked return from DSX$PARSE and the various CLI flags
0000   166 ;                and printed appropriate error messages.
0000   167 ;                Also: changed the DCL-like printout of the command line in error
0000   168 ;                to print the portion of the line that was good. This was done becaus
0000   169 ;                of the strange results from doing it the previous way!!!???
0000   170 ;                Also: added action routines ACT_OPTIONAL, ACT_REQUIRED,
0000   171 ;                ACT_IF_REQUIRED, ACT_IF_FILE_SPEC, ACT_BINARY, and ACT_VERIFY.
0000   172 ;                Also: added recognition of SET BINARY, SET VERIFY, CLEAR BINARY,
0000   173 ;                and CLEAR VERIFY. These are new control flags.
0000   174 ;                Also added [illegible]ANCE action routine. It function exactly opposite of
0000   175 ;                BACKUP.
0000   176 ;
0000   177 ;        46      - Jack [illegible]ry, 13-Nov-1981, Version 6.5
0000   178 ;                Added [illegible]_CALL, CLI$K_RETURN, and CLI$K_BIFS to various places
0000   179 ;                in the parse tree. Dave Butenhof added these new character codes.
0000   180 ;                They are to be used for one-level-deep subroutine calls. Also added
0000   181 ;                SUCCESS and FAILURE action routines to be used in conjunction with
0000   182 ;                the above. Also added a new DS command: EXIT.  Also altered
0000   183 ;                ACT_DO_CMD to avoid dispatching if CLI$L_COMMAND(R2) is 0; this
0000   184 ;                allows ATTACH to 'kill' the command when the /Adapter value is bad.
0000   185 ;                Also added the SHOW SECTIONS command.
0000   186 ;
0000   187 ;        47      - Jack Stansbury, 17-Nov-1981, Version 6.5
0000   188 ;                Changed all BSBW's to JSB's.
0000   189 ;
0000   190 ;        48      - Dave Butenhof, 18-Nov-1981, version 6.5
0000   191 ;                Add DEATTACH command, and typecodes in messages
0000   192 ;
0000   193 ;        49      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000   194 ;                Took out the DSQA constants and the calls to QA_MAIN.
0000   195 ;                It appears these will not be needed for the 6.5 DS version.
```

```
0000   197 ;
0000   198 ;       50      - Dave Butenhof, 04-Feb-1982, Version 6.6
0000   199 ;               Use the new CLI$K_FILE parse code for filename/type parse
0000   200 ;
0000   201 ;       51      - Dave Butenhof, 05-Feb-1982, Version 6.6
0000   202 ;               Add new command SHOW MEMORY.
0000   203 ;
0000   204 ;       52      - Jack Stansbury, 18-Feb-1982, Version 6.6
0000   205 ;               Changed the HELP command parsing so that HSOW will not result
0000   206 ;               in the HELP command looking for DS command SOW! That is, if
0000   207 ;               spaces are not found after the minimum characters for HELP
0000   208 ;               (H in this case), then there must be an EOL. If not EOL, it
0000   209 ;               is an error. Also, I added the VSDP Supervisor command. This
0000   210 ;               is for CRD use. Right now, it simply prints "Command not
0000   211 ;               implemented".
0000   212 ;
0000   213 ;       53      - Dave Butenhof, 09-Mar-1982, version 6.6
0000   214 ;               Change file parsing to accept subdirectories in standalone.
0000   215 ;
0000   216 ;       54      - Dave Butenhof, 10-Mar-1982, version 6.6
0000   217 ;               Correct HELP parsing to accept 'H<CR>' as well as 'H '.
```

```
0000   219 ;
0000   220 ;       55      - Jack Stansbury, 23-Mar-1982, Version 6.7
0000   221 ;               Added some macros (Br_If_Not_...) and took out some .LIST
0000   222 ;               statements.
0000   223 ;
0000   224 ;       56      Jack Stansbury, 10-Apr-1982, Version 6.7
0000   225 ;               Added support for running CRD-AutoTest on-line via   e VSDP
0000   226 ;               CLI command.
0000   227 ;
0000   228 ;       57      Dave Butenhof, 13-Apr-1982, version 6.7
0000   229 ;               Add command to enable/disable generic name enforcement,
0000   230 ;               as backup. 'Set/Clear Enforce'
```

```
0000    232 ;
0000    233 ;       58      Jack Stansbury, 6-May-1982, Version 6.8
0000    234 ;               Added code in DS$Cli to not clear the ^C bit in the DS flags
0000    235 ;               longword if CRD-AutoTest was running. This is to allow ^C's
0000    236 ;               to be caught faster and easier by CRD.
0000    237 ;
0000    238 ;       59      Richard Brown, 19-May-82, Version 6.8
0000    239 ;               Addition of DEBUG command.  This involves placing the command
0000    240 ;               in the command line interpreter decision tree and adding an
0000    241 ;               action routine for the command.
0000    242 ;
0000    243 ;       60      Jack Stansbury, 24-May-1982, Version 6.8
0000    244 ;               Changed the "VSDP" command to "CRD".
0000    245 ;
C000    246 ;       61      Dave Butenhof, 25-May-1982, version 6.8
0000    247 ;               Restore poor SHOW MM command, which got kicked out by
0000    248 ;               SHOW MEMORY (note, however, that SHOW MEMORY did not
0000    249 ;               intend to do this, and should not be accused of bullying).
0000    250 ;
0000    251 ;       62      Richard Brown, 7-July-82, Version 6.8
0000    252 ;               Addition of /DEBUG qualifier to RUN and START commands.
```

```
0000    254 ;
0000    255 ;       63      Jack Stansbury, 15-July-1982, Version 6.9
0000    256 ;               Added code to call CRD Menu Test when the CRD command is typed.
0000    257 ;
0000    258 ;       64      Jack Stansbury, 24-July-1982, Version 6.9
0000    259 ;               Added code (commented out for now) that will check to see if the
0000    260 ;               CRD Menu Test command (CRD) is being called in user mode. If so,
0000    261 ;               it will print an error message. NOTE: this will have to be taken
0000    262 ;               out before the Supervisor is frozen for 6.9!!!!
0000    263 ;
0000    264 ;       65      Jack Stansbury, 8-September-1982, Version 6.9
0000    265 ;               Added three new commands: SET CRDTRACE, CLEAR CRDTRACE, and
0000    266 ;               SHOW CRDTRACE. These three all have to do with a new DSA bit
0000    267 ;               called DSA$V_CRD_Trace. This bit, when set, will cause the CRD
0000    268 ;               routines to print trace and debugging information for the user.
0000    269 ;
0000    270 ;       66      Jack Stansbury, 15-September-1982, Version 6.9
0000    271 ;               Added code in the ACT_CRD routine that will clear the ^C handlers
0000    272 ;               and the ^C DS flag, and disable ^C's.
0000    273 ;
0000    274 ;       67      Jack Stansbury, 23-September-1982, Version 6.9
0000    275 ;               Changed the CRDTRACE qualifier to CRD/TRACE. The command are now:
0000    276 ;               SET CRD/TRACE, CLEAR CRD/TRACE, and SHOW CRD. Also changed the code
0000    277 ;               a bit at the activation of CRD - if CRD's initialization routine
0000    278 ;               returns failure, the DSR$Unload_CRD routine is now called.
0000    279 ;
0000    280 ;       68      Richard Brown, 27-Sep-82, Version 6.9
0000    281 ;               Changed default debugger name from DELTA to VDSDEBUG.
0000    282 ;
0000    283 ;       69      Jack Stansbury, March 3, 1983, Version 6.11
0000    284 ;               Added the SET CRD/DEBUG/TRACE command. This exact syntax
0000    285 ;               is required to minimize the number of people finding out
0000    286 ;               about and using this command. It turns on debugging mode in
0000    287 ;               CRD, and a LOT of output results!
0000    288 ;
0000    289 ;       70      Jack Stansbury, March 7, 1983, Version 6.11
0000    290 ;               Added the SHOW CALLS command. The routine is in the CRDImage module.
0000    291 ;
0000    292 ;       71      Bob Bergazzi    28-March-1983   Version 6.11
0000    293 ;               Added the SET/SHOW PAGE command for video console terminals
0000    294 ;               in standalone mode. Routines are in PRINT.
0000    295 ;
0000    296 ;       72      John Ciukaj     4-April-1983    version 6.11
0000    297 ;               - Changed CRD Command Action routine (ACT_CRD) to
0000    298 ;                 allow online debugging of CRD AUTO Offline
0000    299 ;                 test package and offline initiation of
0000    300 ;                 CRD AUTO Offline by CRD Command.
0000    301 ;               - Changed references to CRD bits in DSA Flags to
0000    302 ;                 distinguish between online and offline.
0000    303 ;
0000    304 ;       73      Bob Bergazzi    8-Apr-1983      Version 6.11
0000    305 ;               Cleared Line_count when we put out a DS>, so that if we are SET PAGE
0000    306 ;               our first screen isn't wasted with stuff we already saw.
0000    307 ;
0000    308 ;       74      Bob Bergazzi    May 16,1983     Version 6.11
0000    309 ;               Changed the order of the .LIB statements.
0000    310 ;
```

```
0000   311 ;        75      Bob Bergazzi    29-Aug-1983     Version 6.13
0000   312 ;                Added the SET MEMORY command.
0000   313 ;
0000   314 ;        76      Bob Bergazzi    20-Sep-1983     Version 6.13
0000   315 ;                Added the /TIME switch to the RUN and START commands.
0000   316 ;                Commented out for release 13 and 14.
0000   317 ;
0000   318 ;        77      Bob Bergazzi    21-Oct-1983     Version 6.13
0000   319 ;                Added the INI PCS command.
0000   320 ;
0000   321 ;        78      Domenic Andella 8-Nov-83        Version 6.14
0000   322 ;                Added branches to tree at GET_FILE_SPEC, and
0000   323 ;                START_AND_RUN routines. These modifications
0000   324 ;                are to allow '$' and '_' to be included in the
0000   325 ;                directory, filename, and file type when using
0000   326 ;                @ script filename, SET LOAD cmd, LOAD cmd, RUN,
0000   327 ;                and DIRECTORY cmd. These changes are necessary
0000   328 ;                for VDS to run under VMS V4.
0000   329 ;
0000   330 ;        79      RE MUSE         8 MAY 1984      Version 7.0
0000   331 ;                allow device names of the form 'node$ggan'
0000   332 ;
0000   333 ;        80      Bob Bergazzi    May 21, 1984    Version 7.0
0000   334 ;                Added /WIDE switch to the DIRECTORY command so
0000   335 ;                that filenames will be displayed one per line.
0000   336 ;                This fixes the problem of truncation of long filenames.
0000   337 ;--
```

```
0000    339          .SBTTL  Libraries, Macros, Equated Symbols
0000    340 ;
0000    341 ; INCLUDE FILES:
0000    342 ;
0000    343
0000    344          .LIBRARY        /SYS$LIBRARY:LIB/       ;                       [40]
0000    345          .Library        /$CRD/                  ;                       [56]
0000    346          .LIBRARY        /$DS/                   ;                       [40]
0000    347          .LIBRARY        /$DIAG/                 ;                       [40]
0000    348 ;
0000    349 ; MACROS:
0000    350 ;
0000    351
0000    352 .MACRO  CASEDEF ACT, N
0000    353          $$N=$$N+1
0000    354          ACT     = N
0000    355          .WORD   ACT_'ACT - 10$
0000    356 .ENDM   CASEDEF
0000    357
0000    358 ;
0000    359 ; EQUATED SYMBOLS:
0000    360 ;
0000    361
0000    362          $DS_CLIDEF                      ; CLI opcode defs
0000    363          $DS_DSADEF                      ; APT mail box defs
0000    364          $DS_DSDEF                       ; Supervisor return codes
0000    365          CLIDEF                          ; CLI data structure
0000    366          DSFDEF                          ; Supervisor internal flags
0000    367          $IPLDEF                         ; IPL level defs
0000    368          $PRDEF                          ; Processor register defs
0000    369          $PSLDEF                         ; PSL field defs
0000    370          $DS_TypeDef                     ; Define type codes             [48]
0000    371          $CRD_Literals                   ; Define CRD equated symbols    [56]
0000    372
```

```
                     0000   374          .SUBTITLE       Work Psect Storage
                     0000   375 ;
                     0000   376 ; OWN STORAGE:
                     0000   377 ;
                 00000000   378          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                     0000   379
                     0000   380 DS$GL_CLIBASE::
                 00000444   0000   381          .BLKB   CLI$K_SIZE
                     0444   382
                     0444   383 Q_GOOD_CMD:                          ; The portion of the command line that  [43]
                 0000044C   0444   384          .BLKQ   1              ;   that is good. This is a descriptor. [43]
                     044C   385
                     044C   386 B_SUCCESS:                          ; Used to store SUCCESS or FAILURE      [46]
                 0000044D   044C   387          .BLKB   1              ; for CLI subroutines                   [46]
```

```
                                044D   389          .SUBTITLE       Data Psect Storage
                                044D   390 ;
                                044D   391 ; STRING STORAGE
                                044D   392 ;
                         00000000   393          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                0000   394
                                0000   395          MODNAM  CLI
                                0004   396
                                0004   397 T_PRGERR:
6D 6D 61 72 67 6F 72 50 20 3F 3F 00'    0004   398          .ASCIC  "?? Programming error!!"                        ;       [43]
   21 21 72 6F 72 72 65 20 67 6E 69     0010
                                16      0004
                                001B   399
                                001B   400 T_ILLCMD:
20 6C 61 67 65 6C 6C 49 20 3F 3F 00'    C01B   401          .ASCIC  "?? Illegal command!/   \!AS\!/"                ;       [43]
20 20 20 2F 21 64 6E 61 6D 6D 6F 63     0027
               2F 21 5C 53 41 21 5C     0033
                                1E      001B
                                003A   402
                                003A   403 T_INCOMPLETE:
65 6C 70 6D 6F 63 6E 49 20 3F 3F 00'    003A   404          .ASCIC  "?? Incomplete command line!/   \!AS\!/"        ;       [43]
6C 20 64 6E 61 6D 6D 6F 63 20 65 74     0046
53 41 21 5C 20 20 20 2F 21 65 6E 69     0052
                  2F 21 5C              005E
                                26      003A
                                0061   405
                                0061   406 T_ILLEFN:                                                    ;       [45]
20 6C 61 67 65 6C 6C 49 20 3F 3F 00'    0061   407          .ASCIC  "?? Illegal event flag number. Must be 1 to 23.!/";     [45]
6E 20 67 61 6C 66 20 74 6E 65 76 65     006D
20 74 73 75 4D 20 2E 72 65 62 6D 75     0079
21 2E 33 32 20 6F 74 20 31 20 65 62     0085
                                2F      0091
                                30      0061
                                0092   408
                                0092   409 T_CONT:
69 75 6E 69 74 6E 6F 43 20 2E 2E 00'    0092   410          .ASCIC  ".. Continuing from !XL!/"                      ;       [43]
21 4C 58 21 20 6D 6F 72 66 20 67 6E     009E
                                2F      00AA
                                18      0092
                                00AB   411
                                00AB   412 T_EMESSG1::
20 64 6E 61 6D 6D 6F 43 20 3F 3F 00'    00AB   413          .ASCIC "?? Command not valid in user mode !/"           ;       [43]
6E 69 20 64 69 6C 61 76 20 74 6F 6E     00B7
21 20 65 64 6F 6D 20 72 65 73 75 20     00C3
                                2F      00CF
                                24      00AB
                                00D0   414
                                00D0   415 T_SHOWMM:
61 6E 61 6D 20 79 72 6F 6D 65 4D 00'    00D0   416          .ASCIC  "Memory management is !AC.!/"                   ;       [43]
41 21 20 73 69 20 74 6E 65 6D 65 67     00DC
                        2F 21 2E 43     00E8
                                1B      00D0
                                00EC   417
                                00EC   418 T_ON:
                     6E 6F 00'  00EC   419          .ASCIC  "on"                                            ;       [43]
                                02      00EC
                                00EF   420
```

```
                                                00EF    421 T_OFF:
                                     66 66 6F 00' 00EF    422         .ASCIC  "off"                                   ;               [43]
                                              03  00EF
                                                00F3    423
                                                00F3    424 GT_Menu_Error::                                         ;               [67]
6E 69 20 72 6F 72 72 45 20 3F 3F 00'            00F3    425         .Ascic  "?? Error in starting the VAX-11/730 CRD MENU TEST!!!/"  ;  [67]
68 74 20 67 6E 69 74 72 61 74 73 20             00FF
30 33 37 2F 31 31 2D 58 41 56 20 65             010B
45 54 20 55 4E 45 4D 20 44 52 43 20             0117
                  2F 21 21 21 54 53             0123
                              35                00F3
                                                0129    426
                                                0129    427 T_User_Mode_Menu_Error:                                 ;               [67]
6E 6E 61 63 20 75 6F 59 20 3F 3F 00'            0129    428         .Ascic  "?? You cannot run VAX-11/730 CRD MENU TEST in user mode!/"  ;  [67]
31 2D 58 41 56 20 6E 75 72 20 74 6F             0135
45 4D 20 44 52 43 20 30 33 37 2F 31             0141
75 20 6E 69 20 54 53 45 54 20 55 4E             014D
      2F 21 65 64 6F 6D 20 72 65 73             0159
                                 39             0129
                                                0163    429
                                                0163    430 T_CRD_Trace_Output:                                     ;               [65]
61 72 54 20 44 52 43 20 65 68 54 00'            0163    431         .Ascic  "The CRD Trace flag is !AC.!/"          ;               [65]
21 20 73 69 20 67 61 6C 66 20 65 63             016F
                     2F 21 2E 43 41             017B
                                 1C             0163
                                                0180    432
```

```
                        .SUBTITLE Commands A-C

                ;+
                ; Command line interpretor decision tree.
                ;-

                ;
                ; Basic commands.
                ;

COMMAND_TREE::
                $DS_CLI CLI$K_SPACE,    0,      100$                    ; Optional <SP>'s

100$:   $DS_CLI <^A"A">,        NOTNUF,   120$                    ; A

                ;
                ; ABORT
                ;

                $DS_CLI CLI$K_STRING, ABORT,    110$,   <'BORT'>        ; ABORT
                $DS_CLI CLI$K_BR,     ENUF,     EOL

                ;
                ; ATTACH
                ;

110$:   $DS_CLI CLI$K_STRING, ATTACH,   ILLEGAL,<'TTACH'>       ; ATTACH
                $DS_CLI CLI$K_EXIT,   ENUF                              ; Enough input gotte

                ;+
                ; The following are allowed as abbreviations for those commands starting       [65]
                ; with the letter "C":                                                         [65]
                ;                                                                             [65]
                ;       CL      -> CLEAR                                                      [65]
                ;       CO      -> CONTINUE                                                   [65]
                ;       CR      -> CRD                                                        [65]
                ;-

120$:   $DS_CLI <^A"C">,        NOTNUF,   140$                    ; C

                ;
                ; CLEAR
                ;

                $DS_CLI CLI$K_STRING, CLEAR,    130$,   <'LEAR'>        ; CLEAR
                $DS_CLI CLI$K_BR,     NOTNUF,   CLEAR_PARAMS            ; Get rest of line

                ;
                ; CONTINUE
                ;

130$:   $DS_CLI CLI$K_STRING, CONTINUE, 135$,   <'ONTINUE'>     ; CONTINUE      [60]
                $DS_CLI CLI$K_BR,     ENUF,     EOL

                ;                                                                             [60]
                ; CRD                                                                         [60]
                ;                                                                             [60]
```

```
01C4    491
01C4    492 135$:   $DS_Cli CLI$K_String, 0,       Illegal,<'RD'>          ; CRD              [65]
01CB    493         $DS_Cli CLI$K_Eol,    CRD,      Illegal                 ; Must be EOL      [65]
```

```
01CF   495          .SubTitle       Directory, Deattach, Deselect, Deposit Commands
01CF   496 ;
01CF   497 ; The commands starting with D.
01CF   498 ; Allow the following as minimums:
01CF   499 ;        D       => DEPOSIT
01CF   500 ;        DI      => DIRECTORY
01CF   501 ;        DEA     => DEATTACH                                                  [48]
01CF   502 ;        DEB     => DEBUG                                                     [59]
01CF   503 ;        DES     => DESELECT
01CF   504 ;
01CF   505
01CF   506 140$:    $DS_CLI <^A'D'>,        NOTNUF,   200$                  ; D
01D3   507
01D3   508 ;
01D3   509 ; DIRECTORY
01D3   510 ;
01D3   511
01D3   512          $DS_CLI CLI$K_STRING, DIRECTORY,150$,   <'IRECTORY'>    ; DIRECTORY
01E0   513          $DS_CLI CLI$K_SLASH,    NOTNUF, 145$                    ; /             [80]
01E4   514          $DS_CLI CLI$K_STRING, DIR_WIDE, BAD_QUALIFIER, <'WIDE'> ; /WIDE         [80]
01ED   515 145$:    $DS_CLI CLI$K_SPACE,    0,      EOL                     ; If not <SP>'s, EOL
01F1   516          $DS_CLI CLI$K_BR,     OPTIONAL, GET_FILE_SPEC           ; Optional file-spec
01F5   517
01F5   518 150$:    $DS_CLI <^A'E'>,        0,      180$                    ; DE
01F9   519          $ds_cli <^A'A'>,        0,      152$                    ; DEA           [59]
01FD   520          $ds_cli cli$k_br,     backup,   156$                    ; Back over 'A' [48]
0201   521 152$:    $DS_CLI <^A'B'>,        0,      155$                    ; DEB           [59]
0205   522          $DS_CLI CLI$K_BR,       BACKUP, 158$                    ; BACK OVER 'B' [59]
0209   523 155$:    $DS_CLI <^A'S'>,        0,      170$                    ; DES           [48]
020D   524          $DS_CLI CLI$K_BR,     BACKUP,   160$                    ; Backup over 'S'
0211   525
0211   526 ;
0211   527 ; DEATTACH
0211   528 ;
0211   529
0211   530 156$:    $ds_cli cli$k_string, deattach, illegal,<'ATTACH'>      ; DEATTACH      [48]
021C   531          $ds_cli cli$k_br,     notnuf,   deattach_qual           ; Get params    [48]
0220   532                                                                                  [59]
0220   533 ;                                                                              [59]
0220   534 ; DEBUG
0220   535 ;
0220   536
0220   537 158$:    $DS_CLI CLI$K_STRING,   DEBUG,  ILLEGAL, <'BUG'>        ; DEBUG         [59]
0228   538          $DS_CLI <^A/=7>,        0,              1159$           ; IF NOT '=', EOL[59
022C   539          $DS_CLI CLI$K_BR,       OPTIONAL,       GET_FILE_SPEC   ; OPT. FILENAME [59]
0230   540 1159$:   $DS_CLI CLI$K_BR,       DEFAULT_DBG,    159$            ; USE DEFLT NAME[59]
0234   541 159$:    $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;               [59]
0238   542
0238   543 ;
0238   544 ; DESELECT
0238   545 ;
0238   546
0238   547 160$:    $DS_CLI CLI$K_STRING, DESELECT, ILLEGAL,<'SELECT'>      ; DESELECT
0243   548          $DS_CLI CLI$K_BR,     NOTNUF,   DESELECT_QUALS          ; Get qualifiers
0247   549
0247   550 170$:    $DS_CLI CLI$K_BR,     BACKUP,   180$                    ; Backup over 'E'
0248   551
```

```
024B   552 180$:   $DS_CLI CLI$K_BR,     BACKUP,   190$                    ; Backup over "D"
024F   553
024F   554 ;
024F   555 ; DEPOSIT
024F   556 ;
024F   557
024F   558 190$:   $DS_CLI CLI$K_STRING, DEPOSIT,  ILLEGAL,<'DEPOSIT'>    ; DEPOSIT
025B   559         $DS_CLI CLI$K_BR,     NOTAUF,   DEPOSIT_PARAMS          ; Get qualifiers
```

```
                        561          .SUBTITLE       Commands Exit, Examine
                        562 ;
                        563 ; The commands starting with E.
                        564 ; Allow the following as minimums:
                        565 ;       E       => EXAMINE
                        566 ;       EXI     => EXIT
                        567 ;
025F                    568
025F                    569 200$:   $DS_CLI <^A''E''>,      NOTNUF,   235$                    ; E
0263                    570         $DS_CLI <^A''X''>,      NOTNUF,   220$                    ; EX
0267                    571         $DS_CLI CLI$K_STRING,   ENUF,     210$,   <'IT'>          ; EXIT
026E                    572
026E                    573 ;
026E                    574 ; EXIT
026E                    575 ;
026E                    576
026E                    577         $DS_CLI CLI$K_BR,       EXIT,     EOL                     ; EXIT <EOL>
0272                    578
0272                    579 210$:   $DS_CLI CLI$K_BR,       BACKUP,   220$                    ; Backup over ''X''
0276                    580
0276                    581 220$:   $DS_CLI CLI$K_BR,       BACKUP,   230$                    ; Backup over ''E''
027A                    582
027A                    583 ;
027A                    584 ; EXAMINE
027A                    585 ;
027A                    586
027A                    587 230$:   $DS_CLI CLI$K_STRING,   EXAMINE,  ILLEGAL,<'EXAMINE'>     ; EXAMINE
0286                    588         $DS_CLI CLI$K_BR,       NOTNUF,   EXAMINE_PARAMS          ; Get qualifiers
```

```
                        590          .SUBTITLE       Help, InitPCS, Load, Next, Run
                        591 ;
                        592 ; HELP
                        593 ;
                        594
028A    595 235$:       $DS_CLI CLI$K_STRING,   0,       240$,    <'HELP'>        ; HELP
0293    596             $DS_CLI CLI$K_EOL,     HELP,     237$                     ; 'H<CR>'
0297    597             $DS_CLI CLI$K_EXIT,    ENUF                               ;
029B    598 237$:       $DS_CLI CLI$K_SPACE,   HELP,     ILLEGAL                  ; Required space(s)
029F    599             $DS_CLI CLI$K_EXIT,    ENUF                               ; Enough input gotte
02A3    600
                        601 ;
                        602 ; INITPCS                                                                  begi
                        603 ;
02A3    604
02A3    605 240$:       $DS_CLI CLI$K_STRING, INITPCS,  245$,    <'INITPCS'>     ; INITPCS
02AF    606             $DS_CLI CLI$K_BR,      ENUF,     EOL                      ;                        end
02B3    607
                        608 ;
                        609 ; LOAD
                        610 ;
02B3    611
02B3    612 245$:       $DS_CLI CLI$K_STRING, LOAD,     250$,    <'LOAD'>        ; LOAD
02BC    613             $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL                  ; <SP>'s req.
02C0    614             $DS_CLI CLI$K_BR,     REQUIRED, GET_FILE_SPEC            ; File-spec required
02C4    615
                        616 ;
                        617 ; NEXT
                        618 ;
02C4    619
02C4    620 250$:       $DS_CLI CLI$K_STRING, NEXT_INST,260$,    <'NEXT'>        ; NEXT
02CD    621             $DS_CLI CLI$K_SPACE,    0,       EOL
02D1    622             $DS_CLI CLI$K_DEC,    DATA,      ILLEGAL                  ; If not dec., ILLEG
02D5    623             $DS_CLI CLI$K_BR,     ENUF,      EOL
02D9    624
                        625 ;
                        626 ; RUN
                        627 ;
02D9    628
02D9    629 260$:       $DS_CLI CLI$K_STRING, RUN,      270$,    <'RUN'>         ; RUN
02E1    630             $DS_CLI CLI$K_BR,     NOTNUF,   START_AND_RUN
```

```
                                .SUBTITLE       Commands Set, Select, Show, Start, Summary, Xdelta, @

                                ;
                                ; SET
                                ;

02E5    270$:   $DS_CLI <^A''S''>,      NOTNUF,   320$                    ; S                 [60]
02E9            $DS_CLI <^A''E''>,      NOTNUF,   290$                    ; SE
02ED            $DS_CLI <^A''T''>,      SET,      280$                    ; SET
02F1            $DS_CLI CLI$K_BR,       NOTNUF,   SET_PARAMS              ; Get parameters

                                ;
                                ; SELECT
                                ;

02F5    280$:   $DS_CLI CLI$K_STRING, SELECT,   ILLEGAL,<'LECT'>          ; SELECT
02FE            $DS_CLI CLI$K_BR,       NOTNUF,   SELECT_QUALS

                                ;
                                ; SHOW
                                ;

0302    290$:   $DS_CLI CLI$K_STRING, SHOW,     300$,   <'HOW'>           ; SHOW
030A            $DS_CLI CLI$K_BR,       NOTNUF,   SHOW_PARAMS

                                ;
                                ; START
                                ;

030E    300$:   $DS_CLI CLI$K_STRING, START,    310$,   <'TART'>          ; START
0317            $DS_CLI CLI$K_BR,       ENUF,     START_AND_RUN

                                ;
                                ; SUMMARY
                                ;

031B    310$:   $DS_CLI CLI$K_STRING, SUMMARY,  ILLEGAL,<'UMMARY'>        ; SUMMARY
0326            $DS_CLI CLI$K_BR,       ENUF,     EOL

                                ;
                                ; XDELTA
                                ;

032A    320$.   $DS_CLI CLI$K_BIF,      IF_XDELTA,330$
032E            $DS_CLI CLI$K_STRING, XDELTA,   330$,   <'XDELTA'>        ; XDELTA
0339            $DS_CLI CLI$K_BR,       ENUF,     EOL

                                ;
                                ; @ file-spec
                                ;

033D    330$:   $DS_CLI <^A'@'>,        SCRIPT,   EOL                     ; @
0341            $DS_CLI CLI$K_SPACE,    NOTNUF,   340$                    ; Optional <SP>'s

0345    340$:   $DS_CLI CLI$K_BR        REQUIRED, GET_FILE_SPEC           ; File-spec required
```

```
                                .SUBTITLE       End of Line Processing
                                ;
                                ; End of line.
                                ; If really EOL, return SUCCESSfully. If not, advance past the next character
                                ; (the one that is illegal), save the context at this point, and return.
                                ;
                                
0349    692 EOL:    $DS_CLI CLI$K_EOL,      0,      100$                    ; If not EOL, branch
034D    693         $DS_CLI CLI$K_EXIT                                      ; Exit successfully
0351    694
```

```
                0351    696 100$:   $DS_CLI CLI$K_SPACE,    0,      110$                    ; Jump past spaces
```

```
                     0355    698
                     0355    699 110$:   $DS_CLI CLI$K_BR,     ADVANCE,  120$          ; Jump past next sym
                     0355    700                                                        ; that was illegal
                     0359    701
                     0359    702 120$:   $DS_CLI CLI$K_BR,     ILL_CMD,  130$          ; Save current point
                     035D    703                                                        ; in the line
                     035D    704
                     035D    705 130$:   $DS_CLI CLI$K_ERROR                           ; Indicate ILLEGAL
```

```
                        707          .SUBTITLE       Illegal Command Processing
                        708 ;
                        709 ; Errors.
                        710 ;
                        711
                        712 ILLEGAL:
                        713
                        714          ;
                        715          ; Illegal command error:
                        716          ;
                        717          ; This is called with the parse character pointer pointing to the illegal
                        718          ; character. This will usually be a character within an unrecognized
                        719          ; qualifier or parameter such as the "X" in "SET WIDTX". It is possible
                        720          ; for this to be called when a command is actually INCOMPLETE. This
                        721          ; happens, for example, in recognizing "SET BREAKPOINTS" but failing
                        722          ; to see any (required) spaces after the "BREAKPOINTS". If no spaces
                        723          ; are seen, ILLEGAL is called even though the next symbol could
                        724          ; actually be EOL (implying INCOMPLETE). If it is not an EOL, then there
                        725          ; is an illegal character (e.g., the "X" above) and the command is
                        726          ; ILLEGAL. If it is an EOL, then the required address is missing and the
                        727          ; command is INCOMPLETE. Therefore, EOL is checked below to make a
                        728          ; determination about the kind of error that has occurred.
                        729          ;
                        730
0361    731          $DS_CLI CLI$K_BR,     INC_CMD, 100$                          ; Save current point
0365    732                                                                       ; in the line. Assum
0365    733                                                                       ; the line is INCOMP
0365    734
0365    735 100$:    $DS_CLI CLI$K_EOL,      0,      110$                          ; If not EOL, branch
0369    736          $DS_CLI CLI$K_ERROR                                        ; If EOL, exit w/INC
036D    737
036D    738          ;
036D    739          ; Not really EOL, somthing illegal is there. First, get rid of any spaces
036D    740          ; that are there and then advance past the illegal character.
036D    741          ;
036D    742
036D    743 110$:    $DS_CLI CLI$K_SPACE,    0,      115$                          ; Jump past any spac
0371    744
0371    745 115$:    $DS_CLI CLI$K_BR,     ADVANCE,  120$                          ; Jump past illegal
0375    746
0375    747          ;
0375    748          ; Establish new context point one character past start of illegal portion.
0375    749          ; Exit with error.
0375    750          ;
0375    751
0375    752 120$:    $DS_CLI CLI$K_ERROR,  ILL_CMD                                ; Print error mess.
```

```
                     0379   754          .SUBTITLE       Incomplete Command Processing
                     0379   755          ;
                     0379   756          ; Incomplete command error:
                     0379   757          ;
                     0379   758
                     0379   759 INCOMPLETE:
                     0379   760          ;
                     0379   761          ; We have an INCOMPLETE command line.
                     0379   762          ;
                     0379   763
                     0379   764          $DS_CLI CLI$K_ERROR,  INC_CMD                    ; Print error mess.
```

```
                        .SUBTITLE       Bad List Processing
                        ;
                        ; Bad list element error:
                        ;

                        BAD_LIST:

                        ;
                        ; We have a bad list. This is branched to at the end of a
                        ; name recognition list if a name is unrecognized. This
                        ; checks for cases of
                        ;       ,,      => INCOMPLETE
                        ;       ,<EOL> => INCOMPLETE
                        ;       ,???    => ILLEGAL
                        ;

                        $DS_CLI CLI$K_COMMA,    0,      100$                    ; If not comma, bran
                        $DS_CLI CLI$K_BR,       0,      INCOMPLETE              ; ,,

                        100$:   $DS_CLI CLI$K_BR,     INC_CMD,  110$            ; Assume INCOMPLETE

                        110$:   $DS_CLI CLI$K_EOL,      0,      120$            ; If not EOL, branch
                                $DS_CLI CLI$K_ERROR                             ; ,<EOL>

                        120$:   $DS_CLI CLI$K_SPACE,    0,      130$            ; Jump past any spac

                        130$:   $DS_CLI CLI$K_BR,     ADVANCE,  140$            ; Jump past illegal

                        140$:   $DS_CLI CLI$K_ERROR,  ILL_CMD                   ; ILLEGAL
```

```
                        796          .SUBTITLE       Bad Qualifier Processing
                        797          ;
                        798          ; Bad qualifier error:
                        799          ;
                        800
                        801 BAD_QUALIFIER:
                        802          ;
                        803          ; We have a bad qualifier. This is branched to at the end of a
                        804          ; qualifier recognition sub-tree if a qualifier is not
                        805          ; recognized. This checks for cases of
                        806          ;       //      => INCOMPLETE
                        807          ;       /<EOL> => INCOMPLETE
                        808          ;       /???    => ILLEGAL
                        809          ;
                        810
                        811          $DS_CLI CLI$K_SLASH,    0,      100$                    ; If not /, branch
                        812          $DS_CLI CLI$K_BR,       0,      INCOMPLETE              ; //
                        813
                        814 100$:    $DS_CLI CLI$K_BR,     INC_CMD,  110$                    ; Assume INCOMPLETE
                        815
                        816 110$:    $DS_CLI CLI$K_EOL,      0,      120$                    ; /???
                        817          $DS_CLI CLI$K_ERROR                                     ; /<EOL>
                        818
                        819 120$:    $DS_CLI CLI$K_SPACE,    0,      130$                    ; Jump past any spac
                        820
                        821 130$:    $DS_CLI CLI$K_BR,     ADVANCE,  140$                    ; Jump past illegal
                        822
                        823 140$:    $DS_CLI CLI$K_ERROR,  ILL_CMD                           ; ILLEGAL.
```

```
                        .SUBTITLE       Get Generic Device Name
                        ;
                        ; Finish scanning SELECT... and DESELECT... and SHOW DEVICE...
                        ; Each one allows a list of generic device names. The list is composed
                        ; of names separated by commas. Spaces may appear on either side of
                        ; the commas.
                        ; Store the generic device name in the CLI$Q_FILE quadword as a
                        ; descriptor.
                        ;

                        GET_DEV_NAME:
                                $DS_CLI CLI$K_SPACE,    0,      100$                    ; Optional <SP>'s

                        100$:   $DS_CLI CLI$K_BR,       FILESPEC, 110$

                        110$:   $DS_CLI CLI$K_ALNUM,    FILEND,   140$                  ; Alpha-nums
                                $DS_CLI <^A"$">,        FILEND,   116$                  ; Optional $    [79]
                                $DS_CLI CLI$K_BR,         0,      110$
                        116$:   $DS_CLI <^A":">,          0,      120$                  ; Optional :

                        120$:   $DS_CLI CLI$K_COMMA,    DO_CMD,   130$                  ; Do command for
                                                                                        ;   each dev. name
                                $DS_CLI CLI$K_BR,         0,      100$                  ; Get next one

                        130$:   $DS_CLI CLI$K_BR,       ENUF,     EOL                   ; Signal enough inpu

                        140$:   $DS_CLI CLI$K_BIF,      IF_REQUIRED,INCOMPLETE          ; If name req., bran
                                $DS_CLI CLI$K_BR,       ENUF,     EOL                   ; Assume EOL
```

```
                        .Subtitle       Deattach Processing
                ;
                ; The Deattach command requires a device name and a /Adapter qualifier.
                ; It will accept only one device name and one /Adapter qualifier.
                ;                                                                       [48]
deattach_qual:                                                                          ;
        $DS_CLI CLI$K_SLASH,  NOTNUF,   110$                                            ; If not /, get name

        $DS_CLI CLI$K_STRING,   0,      BAD_QUALIFIER,<'ADAPTER'>; / ADAPTER
        $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                                        ; Required :
        $DS_CLI CLI$K_BR,     BEGINADAPTER,100$                                         ; Start of name

100$:   $DS_CLI CLI$K_ALNUM,  ADAPTER,  ILLEGAL                                         ; Required alpha-num
        $DS_CLI <^A":">,        0,      110$                                            ; Optional :

110$:   $ds_cli cli$k_bif, if_file_spec,140$                                            ; End if device
        $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                                         ; Required <SP>'s
        $ds_cli cli$k_br,     filespec, 120$                                            ; Initialize
120$:   $ds_cli cli$k_alnum,  filend,   incomplete                                      ; Device name
        $ds_cli <^A"$">,      filend,   125$                                            ; Optional '$
        $ds_cli cli$k_br,       0,      120$                                            ;
125$:   $ds_cli <^A":">,        0,      130$                                            ; Optional ':'
130$:   $ds_cli cli$k_bif,  if_adapter, 140$                                            ; End if adapter
        $ds_cli cli$k_br,     notnuf,   deattach_qual                                   ; Get adapter
140$:   $ds_cli cli$k_bif,  if_adapter, eol                                             ; EOL
        $ds_cli cli$k_br,       0,      incomplete                                      ; No good
```

```
                                        .SUBTITLE       Deselect and Select Processing
                                ;
                                ; Get /ADAPTER qualifier for the SELECT and DESELECT commands.
                                ; Both accept a /, followed by ADAPTER, followed by (and required) a : or =,
                                ; followed by (and required) a name, followed optionally by a colon. Only one
                                ; such qualifier is allowed. Spaces must appear before the generic device name.
                                ;

                        SELECT_QUALS:
                        DESELECT_QUALS:
                                $DS_CLI CLI$K_SLASH,  NOTNUF,   110$                    ; If not /, get name

                                $DS_CLI CLI$K_STRING,   0,      BAD_QUALIFIER,<'ADAPTER'>; / ADAPTER
                                $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                 ; Required :
                                $DS_CLI CLI$K_BR,     BEGINADAPTER,100$                 ; Start of name

                        100$:   $DS_CLI CLI$K_ALNUM,  ADAPTER,  ILLEGAL                 ; Required alpha-num
                                $DS_CLI <^A":">,        0,      110$                    ; Optional :

                        110$:   $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s
                                $DS_CLI CLI$K_BR,     REQUIRED, GET_DEV_NAME            ; Get req. name
```

```
                        045D   904          .SUBTITLE       Get a File Specification
                        045D   905 ;
                        045D   906 ; File specification for a script file name, SET LOAD command,
                        045D   907 ; LOAD command, and DIRECTORY command. The file-spec must be
                        045D   908 ; followed by an EOL. At the end of getting the file-spec,
                        045D   909 ; check to see if a file-spec was required. If it was
                        045D   910 ; required, and none was gotten, it is an INCOMPLETE error.
                        045D   911 ;
                        045D   912
                        045D   913 GET_FILE_SPEC:
                        045D   914          $DS_CLI CLI$K_BR,      FILESPEC, 100$
                        0461   915
                        0461   916 100$:    $DS_CLI CLI$K_ALPHA,   FILEND,   120$                  ; Alpha char(s)
                        0465   917          $DS_CLI CLI$K_SYMBOL,  FILEND,   110$                  ; Alpha-nums      [78
                        0469   918
                        0469   919 110$:    $DS_CLI <^A":">,       FILEND,   160$                  ; :
                        046D   920
                        046D   921 120$:    $DS_CLI <^A"[">,         0,      160$                  ; [
                        0471   922          $DS_CLI CLI$K_OCT,       0,      130$                  ; Look for octal-
                        0475   923                                                                 ;   range digits
                        0475   924          $DS_CLI CLI$K_COMMA,     0,      130$                  ; Is it [0,0] form?
                        0479   925          $DS_CLI CLI$K_OCT,       0,      INCOMPLETE            ; Error if "[n,"
                        047D   926                                                                 ;   not finished
                        047D   927          $DS_CLI CLI$K_BR,        0,      150$                  ; Look for ']'
                        0481   928
                        0481   929 130$:    $DS_CLI CLI$K_ALNUM,     0,      150$                  ; ALLOW ALPHA-NUM
                        0485   930
                        0485   931 132$:    $DS_CLI CLI$K_ALNUM,     0,      135$                  ; ALLOW ALPHA-NUM
                        0489   932 135$:    $DS_CLI <^A"$">,         0,      137$                  ; ALLOW '$'
                        048D   933          $DS_CLI CLI$K_BR,        0,      132$                  ; GO LOOK FOR MORE A
                        0491   934 137$:    $DS_CLI <^A"_">,         0,      140$                  ; ALLOW '_'
                        0495   935          $DS_CLI CLI$R_BR,        0,      132$                  ; GO LOOK FOR MORE A
                        0499   936
                        0499   937 140$:    $DS_CLI <^A".">,         0,      150$                  ; .               [38]
                        049D   938          $DS_CLI CLI$K_ALNUM,     0,      INCOMPLETE            ;                 [78]
                        04A1   939
                        04A1   940 142$:    $DS_CLI CLI$K_ALNUM,     0,      145$                  ; ALLOW ALPHA-NUM
                        04A5   941 145$:    $DS_CLI <^A"$">,         0,      147$                  ; ALLOW '$'
                        04A9   942          $DS_CLI CLI$K_BR,        0,      142$                  ; GO LOOK FOR MORE A
                        04AD   943 147$:    $DS_CLI <^A"_">,         0,      148$                  ; ALLOW '_'
                        04B1   944          $DS_CLI CLI$R_BR,        0,      142$                  ; GO LOOK FOR MORE A
                        04B5   945
                        04B5   946 148$:    $DS_CLI <^A"]">,       FILEND,   140$                  ; ]
                        04B9   947          $DS_CLI CLI$K_BR,        0,      160$                  ;                 [38]
                        04BD   948
                        04BD   949 150$:    $DS_CLI <^A"]">,       FILEND,   INCOMPLETE            ; ]
                        04C1   950
                        04C1   951 160$:    $DS_CLI CLI$K_FILE,    FILEND,   170$                  ;
                        04C5   952
                        04C5   953 162$:    $DS_CLI CLI$K_ALNUM,     FILEND, 165$                  ; ALLOW ALPHA-NUM
                        04C9   954 165$:    $DS_CLI <^A"$">,         FILEND, 167$                  ; ALLOW '$'
                        04CD   955          $DS_CLI CLI$K_BR,        0,      162$                  ; GO LOOK FOR MORE A
                        04D1   956 167$:    $DS_CLI <^A"_">,         FILEND, 170$                  ; ALLOW '_'
                        04D5   957          $DS_CLI CLI$R_BR,        0,      162$                  ; GO LOOK FOR MORE A
                        04D9   958
                        04D9   959 170$:    $DS_CLI <^A".">,       FILEND,   180$                  ; .
                        04DD   960          $DS_CLI CLI$K_FILE,    FILEND,   180$                  ;
```

```
04E1   961
04E1   962 172$:   $DS_CLI CLI$K_ALNUM,    FILEND, 175$                    ; ALLOW ALPHA-NUM
04E5   963 175$:   $DS_CLI <^A"$">,        FILEND, 177$                    ; ALLOW '$'
04E9   964         $DS_CLI CLI$K_BR,       0,      172$                    ; GO LOOK FOR MORE A
04ED   965 177$:   $DS_CLI <^A" ">,        FILEND, 180$                    ; ALLOW ' '
04F1   966         $DS_CLI CLI$K_BR,       0,      172$                    ; GO LOOK FOR MORE A
04F5   967
04F5   968 180$:   $DS_CLI <^A";">,      FILEND,   190$                    ; ";"
04F9   969         $DS_CLI CLI$K_DEC,    FILEND,   190$                    ; Decimal version
04FD   970 190$:
04FD   971         ;+
04FD   972         ; Check if the file-spec is present.
04FD   973         ; If present,
04FD   974         ;       If it was required,
04FD   975         ;               INCOMPLETE
04FD   976         ;       else
04FD   977         ;               EOL
04FD   978         ; else
04FD   979         ;       EOL
04FD   980         ;-
04FD   981
04FD   982         $DS_CLI CLI$K_BIF,    IF_FILE_SPEC,200$                 ; If file-spec, bran
0501   983         $DS_CLI CLI$K_BIF,    IF_REQUIRED,INCOMPLETE            ; If required, branc
0505   984
0505   985 200$:   $DS_CLI CLI$K_BR,     ENUF,     EOL                     ; Enough input
```

```
0509   987          .SUBTITLE       Start and Run Processing
0509   988 ;+
0509   989 ; This is for both the START and the RUN commands. The file-spec
0509   990 ; part should be a mirror image of the above tree portion.
0509   991 ; The START does not allow a file-spec, but the RUN does.
0509   992 ; Both allow the same qualifiers. In here, get both the file-spec
0509   993 ; (only if it is a RUN command AND if a file-spec has not already
0509   994 ; been given) and the qualifiers. Allow qualifiers both before and
0509   995 ; after the file-spec. Allow spaces before and after the /'s.
0509   996 ; Do not allow only a "/" (with no qualifier name).
0509   997 ;
0509   998 ; ALGORITHM:
0509   999 ;       LOOP
0509  1000 ;               IF SLASH,
C509  1001 ;                       GET A QUALIFIER
0509  1002 ;                       CONTINUE LOOP
0509  1003 ;               ELSE
0509  1004 ;                       IF RUN COMMAND AND NO FILE-SPEC GOTTEN YET,
0509  1005 ;                               GET FILE-SPEC
0509  1006 ;                               IF FILE-SPEC GOTTEN,
0509  1007 ;                                       CONTINUE LOOP
0509  1008 ;                               ELSE
0509  1009 ;                                       ; No file-spec given, but
0509  1010 ;                                       ; the RUN command requires
0509  1011 ;                                       ; one.
0509  1012 ;                                       INCOMPLETE COMMAND LINE
0509  1013 ;                               FI
0509  1014 ;                       ELSE
0509  1015 ;                               ; Better be end-of-line
0509  1016 ;                               ENUF, EOL
0509  1017 ;                       FI
0509  1018 ;               FI
0509  1019 ;       END LOOP
0509  1020 ;-
0509  1021
0509  1022 START_AND_RUN:
0509  1023
0509  1024          $DS_CLI CLI$K_SLASH,    0,      100$                    ; If no /, branch
050D  1025          $DS_CLI CLI$K_BR,       NOTNUF, START_RUN_QUALIFIERS    ; If /, branch
0511  1026
0511  1027 100$:    $DS_CLI CLI$K_BIF,      IF_RUN, 110$                    ; If RUN and no
0515  1028                                                                  ; file-spec yet, go
0515  1029          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ; If START or (RUN
0519  1030                                                                  ; and file-spec
0519  1031                                                                  ; gotten), then EOL.
```

```
0519  1033
0519  1034          ;+
0519  1035          ; The below portion is only executed for the RUN command,
0519  1036          ; and then it is executed only once. If a file-spec is found,
0519  1037          ; this branches back up to START_AND_RUN to get more qualifiers.
0519  1038          ; If no file-spec is found, this is an INCOMPLETE error.
0519  1039          ; Note that spaces must always appear before the file-spec.
0519  1040          ;-
0519  1041
0519  1042 110$:    $DS_CLI CLI$K_SPACE,    0,       ILLEGAL                  ; Required <SP>'s
051D  1043
051D  1044 120$:    $DS_CLI CLI$K_BR,       FILESPEC, 130$                    ; Save start of
0521  1045                                                                    ; file-spec.
0521  1046 130$:    $DS_CLI CLI$K_ALPHA,    FILEND,   150$                    ; Alpha-char(s)
0525  1047          $DS_CLI CLI$K_SYMBOL,   FILEND,   140$                    ; Alpha-num, $, and
0529  1048
0529  1049 140$:    $DS_CLI <^A":">,        FILEND,   200$                    ; :
052D  1050
052D  1051 150$:    $DS_CLI <^A"[">,          0,      190$                    ; [
0531  1052          $DS_CLI CLI$K_OCT,        0,      160$                    ; Look for octal-
                                                                              ;   range digits
0535  1053
0535  1054          $DS_CLI CLI$K_COMMA,      0,      160$                    ; Is it [0,0] form?
0539  1055          $DS_CLI CLI$K_OCT,        0,      INCOMPLETE              ; Error if "[n,"
                                                                              ;   not finished
053D  1056
053D  1057          $DS_CLI CLI$K_BR,         0,      180$                    ; Look for ']'
0541  1058
0541  1059 160$:    $DS_CLI CLI$K_ALNUM,      0,      180$                    ; ALLOW ALPHA-NUM
0545  1060
0545  1061 162$:    $DS_CLI CLI$K_ALNUM,      0,      165$                    ; ALLOW ALPHA-NUM
0549  1062 165$:    $DS_CLI <^A"$">,          0,      167$                    ; ALLOW '$'
054D  1063          $DS_CLI CLI$K_BR,         0,      162$                    ; GO LOOK FOR MORE A
0551  1064 167$:    $DS_CLI <^A"_">,          0,      170$                    ; ALLOW '_'
0555  1065          $DS_CLI CLI$K_BR,         0,      162$                    ; GO LOOK FOR MORE A
0559  1066
0559  1067 170$:    $DS_CLI <^A".">,          0,      180$                    ; .                 [78]
055D  1068          $DS_CLI CLI$K_ALNUM,      0,      INCOMPLETE              ;                   [78]
0561  1069
0561  1070 172$:    $DS_CLI CLI$K_ALNUM,      0,      175$                    ; ALLOW ALPHA-NUM
0565  1071 175$:    $DS_CLI <^A"$">,          0,      177$                    ; ALLOW '$'
0569  1072          $DS_CLI CLI$K_BR,         0,      172$                    ; GO LOOK FOR MORE A
056D  1073 177$:    $DS_CLI <^A"_">,          0,      178$                    ; ALLOW '_'
0571  1074          $DS_CLI CLI$K_BR,         0,      172$                    ; GO LOOK FOR MORE A
0575  1075
0575  1076 178$:    $DS_CLI <^A"]">,        FILEND,   170$                    ; ]                 [78]
0579  1077          $DS_CLI CLI$K_BR,         0,      190$                    ;                   [38]
057D  1078
057D  1079 180$:    $DS_CLI <^A"]">,        FILEND,   INCOMPLETE              ; ]
0581  1080
0581  1081 190$:    $DS_CLI CLI$K_ALNUM,    FILEND,   200$                    ;
0585  1082
0585  1083 192$:    $DS_CLI CLI$K_ALNUM,      FILEND, 195$                    ; ALLOW ALPHA-NUM
0589  1084 195$:    $DS_CLI <^A"$">,          FILEND, 197$                    ; ALLOW '$'
058D  1085          $DS_CLI CLI$K_BR,         0,      192$                    ; GO LOOK FOR MORE A
0591  1086 197$:    $DS_CLI <^A"_">,          FILEND, 200$                    ; ALLOW '_'
0595  1087          $DS_CLI CLI$K_BR,         0,      192$                    ; GO LOOK FOR MORE A
0599  1088
0599  1089 200$:    $DS_CLI <^A".">,        FILEND,   210$                    ; .
```

```
059D  1090          $DS_CLI CLI$K_ALNUM,  FILEND,   210$                    ;
05A1  1091
05A1  1092 202$:    $DS_CLI CLI$K_ALNUM,    FILEND, 205$                    ; ALLOW ALPHA-NUM
05A5  1093 205$:    $DS_CLI <^A"$">,        FILEND, 207$                    ; ALLOW '$'
05A9  1094          $DS_CLI CLI$K_BR,       0,      202$                    ; GO LOOK FOR MORE A
05AD  1095 207$:    $DS_CLI <^A"_">,        FILEND, 210$                    ; ALLOW '_'
05B1  1096          $DS_CLI CLI$K_BR,       0,      202$                    ; GO LOOK FOR MORE A
05B5  1097
05B5  1098 210$:    $DS_CLI <^A":">,      FILEND,   220$                    ; ":"
05B9  1099          $DS_CLI CLI$K_DEC,    FILEND,   220$                    ; Decimal version
05BD  1100
05BD  1101 220$:    $DS_CLI CLI$K_BIF,    IF_FILE_SPEC,START_AND_RUN        ; If file-spec
05C1  1102                                                                  ; gotten, branch
05C1  1103          $DS_CLI CLI$K_BR,       0,      INCOMPLETE              ; If not file-spec
05C5  1104                                                                  ; gotten, INCOMPLETE
05C5  1105                                                                  ; for RUN command.
```

```
                        05C5  1107          .SUBTITLE       Start and Run Qualifiers
                        05C5  1108 ;
                        05C5  1109 ; Qualifiers for the START and RUN commands.
                        05C5  1110 ; This allows optional spaces before and after the :'s and the /'s.
                        05C5  1111 ;
                        05C5  1112
                        05C5  1113 START_RUN_QUALIFIERS:
                        05C5  1114                                                                              [62]
                        05C5  1115 ;                                                                            [62]
                        05C5  1116 ; START/DEBUG                                                                [62]
                        05C5  1117 ;
                        05C5  1118
                        05C5  1119          $DS_CLI CLI$K_STRING, DEBUG_SWITCH, 90$, <'DEBUG'>      ; /DEBUG       [62]
                        05CF  1120          $DS_CLI CLI$K_BR,       0,      START_AND_RUN           ; NEXT QUALIFIER[62]
                        05D3  1121
                        05D3  1122 ;
                        05D3  1123 ; START/PASSES:x
                        05D3  1124 ;
                        05D3  1125
                        05D3  1126 90$:                                                         ;               [62]
                        05D3  1127          $DS_CLI CLI$K_STRING, NOTNUF,   100$,   <'PASSES'>      ; /PASSES      [42]
                        05DE  1128          $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                 ; : or =
                        05E2  1129          $DS_CLI CLI$K_DEC,    PASS,     ILLEGAL                 ; /PASSES:decmal
                        05E6  1130          $DS_CLI CLI$K_BR,       0,      START_AND_RUN
                        05EA  1131
                        05EA  1132 ;
                        05EA  1133 ; START/QA
                        05EA  1134 ;
                        05EA  1135
                        05EA  1136 100$:    $DS_CLI CLI$K_STRING  QA,       110$,   <'QA'>          ; /QA          [42]
                        05F1  1137          $DS_CLI CLI$K_BR        0,      START_AND_RUN           ;               [42]
```

```
                     1139 ;
                     1140 ; START/SECTION:name
                     1141 ;
                     1142
05F5  1143 110$:   $DS_CLI <^A'S'>,      NOTNUF,   125$                      ; S
05F9  1144
05F9  1145         $DS_CLI CLI$K_STRING, NOTNUF,   120$,   <'ECTION'>       ; /SECTION
0604  1146         $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                   ; : or =
0608  1147         $DS_CLI CLI$K_SYMBOL, SECTION,  ILLEGAL                   ;                    [37]
060C  1148         $DS_CLI CLI$K_BR,       0,      START_AND_RUN             ;                    [42]
0610  1149 ;
0610  1150 ; START/SUBTEST:x
0610  1151 ;
0610  1152
C610  1153 120$:   $DS_CLI CLI$K_STRING, NOTNUF,   BAD_QUALIFIER,<'UBTEST'>; /SUBTEST
0618  1154         $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                   ; : or =
061F  1155         $DS_CLI CLI$K_DEC,    SUBTEST,  ILLEGAL
0623  1156         $DS_CLI CLI$K_BR,       0,      START_AND_RUN             ;                    [42]
0627  1157
0627  1158 ;
0627  1159 ; START/TEST:x
0627  1160 ; START/TEST:x:y
0627  1161 ;
0627  1162
0627  1163 ;125$:  $DS_CLI <^A'T'>,      NOTNUF,   BAD_QUALIFIER           ; T                  [37]
0627  1164 ;
0627  1165 ;       $DS_CLI CLI$K_STRING, NOTNUF,   130$,   <'EST'>          ; /TEST              [37]
0627  1166
0627  1167 125$:   $DS_CLI CLI$K_STRING, NOTNUF,   BAD_QUALIFIER,  <'TEST'>; /TEST              [37]
0630  1168         $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                   ; : or =
0634  1169         $DS_CLI CLI$K_DEC,    TEST,     ILLEGAL                   ; /TEST:x
0638  1170         $DS_CLI CLI$K_VALUE,  NOTNUF,   145$                      ; : or =
063C  1171         $DS_CLI CLI$K_DEC,    LAST,     ILLEGAL                   ; /TEST:x:y
0640  1172         $DS_CLI CLI$K_BR,     0,        START_AND_RUN             ;
0644  1173
0644  1174 ;
0644  1175 ; START/TIME:dddd-hh:mm:ss.cc                                                          [76]
0644  1176 ;
0644  1177 ;
0644  1178 ;130$:  $DS_CLI CLI$K_STRING, NOTNUF,   BAD_QUALIFIER,<'IME'>   ; /TIME              [76]
0644  1179 ;       $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                   ; : or =             [76]
0644  1180 ;       $DS_CLI CLI$K_BR,     BEGINTIME, 131$                     ; Mark beginning of
0644  1181 ;131$:  $DS_CLI CLI$K_DEC,    0,        133$                     ; dddd               [76]
0644  1182 ;       $DS_CLI <^A'-'>,      0,        133$                     ; SP                 [76]
0644  1183 ;132$:  $DS_CLI CLI$K_DEC,    0,        133$                     ; hh                 [76]
0644  1184 ;133$:  $DS_CLI CLI$K_VALUE,  0,        134$                     ; : or =             [76]
0644  1185 ;134$:  $DS_CLI CLI$K_DEC,    0,        135$                     ; mm                 [76]
0644  1186 ;135$:  $DS_CLI CLI$K_VALUE,  0,        136$                     ; : or =             [76]
0644  1187 ;136$:  $DS_CLI CLI$K_DEC,    0,        137$                     ; ss                 [76]
0644  1188 ;137$:  $DS_CLI <^A'.'>,      0,        138$                     ; .                  [76]
0644  1189 ;138$:  $DS_CLI CLI$K_DEC,    0,        139$                     ; cc                 [76]
0644  1190 ;139$:  $DS_CLI CLI$K_BIF,    TIME,     ILLEGAL                  ; Check the string [
0644  1191 ;
0644  1192 145$:   $DS_CLI CLI$K_BR,       0,      START_AND_RUN             ;                    [42]
```

```
                        .SUBTITLE       Show Parameters and Qualifiers

; SHOW parameters.
;
;       SHOW BASE
;       SHOW BREAKPOINTS
;       SHOW DEFAULTS
;       SHOW CALLS                                                              [70]
;       SHOW CRD                                                                [67]
;       SHOW DEVICE
;       SHOW DEVICE/BRIEF
;       SHOW EVENT FLAGS
;       SHOW EVENT
;       SHOW FLAGS
;       SHOW LOAD
;       SHOW MEMORY [/MAP][/BUFFER][/DATA_STRUCTURE][/ALL]
;       SHOW MM
;       SHOW PAGE
;       SHOW QA...
;       SHOW SECTIONS
;       SHOW SELECTED
;       SHOW SELECTED/BRIEF
;       SHOW STATUS
;       SHOW SUPPORT
;       SHOW WIDTH
;       SHOW/BRIEF DEVICE
;       SHOW/BRIEF SELECTED
;

SHOW_PARAMS:
        $DS_CLI CLI$K_SPACE,    0,      310$                    ; If no <SP>'s, chec
        $DS_CLI <^A'B'>,        NOTNUF, 110$                    ; B

;
; SHOW BASE
;

        $DS_CLI CLI$K_STRING, BASE,     100$,   <'ASE'>         ; BASE
        $DS_CLI CLI$K_BR,       ENUF,   EOL

;
; SHOW BREAKPOINTS
;

100$:   $DS_CLI CLI$K_STRING, BREAK,    ILLEGAL,<'REAKPOINTS'>  ; BREAKPOINTS
        $DS_CLI CLI$K_BR,       ENUF,   EOL

;+
; SHOW CRD                                                                      [67]
;-

110$:   $DS_Cli <^A'C'>,        NotNuf, 115$                    ; SHOW C        [70]
        $DS_Cli CLI$K_String, 0,        112$,   <'RD'>          ; SHOW CRD      [70]
        $DS_Cli CLI$K_Eol,      Show_CRD_Trace, Illegal         ; Must end EOL  [65]
        $DS_Cli CLI$K_Br,       Enuf,   Eol                     ;               [65]
```

```
0682  1251 112$:   $DS_Cli CLI$K_String, 0,         Illegal,<'ALLS'>        ; SHOW CALLS     [70]
068B  1252         $DS_Cli CLI$K_Eol,    Show_Calls,Illegal                 ; Must end EOL   [70]
068F  1253         $DS_Cli CLI$K_Br,     Enuf,      Eol                     ;                [70]
0693  1254
0693  1255 ;+
0693  1256 ; SHOW D . . .
0693  1257 ;-
0693  1258
0693  1259 115$:   $DS_CLI <^A'D'>,      NOTNUF,    150$                    ; D
0697  1260         $DS_CLI <^A'E'>,      NOTNUF,    ILLEGAL                 ; DE
069B  1261
069B  1262 ;
069B  1263 ; SHOW DEFAULTS
069B  1264 ;
C69B  1265
069B  1266         $DS_CLI CLI$K_STRING, DEFAULT,   120$,    <'FAULTS'>     ; DEFAULT
06A6  1267         $DS_CLI CLI$K_BR,     ENUF,      EOL
```

```
                        06AA  1269 ;
                        06AA  1270 ; SHOW DEVICE
                        06AA  1271 ; SHOW DEVICE                          dev_name_1.dev_name_2,...
                        06AA  1272 ; SHOW DEVICE/ADAPTER:name             dev_name_1.dev_name_2,...
                        06AA  1273 ; SHOW DEVICE/BRIEF                    dev_name_1.dev_name_2,...
                        06AA  1274 ; SHOW DEVICE/ADAPTER:name/BRIEF       dev_name_1.dev_name_2,...
                        06AA  1275 ;
                        06AA  1276 ; Note: if the /ADAPTER:name qualifier appears more than once
                        06AA  1277 ; on the command line, the LAST name given is the one used.
                        06AA  1278 ; Spaces may appear on either side of the /'s, ,'s, and :'s.
                        06AA  1279 ;
                        06AA  1280
                        06AA  1281 120$:   $DS_CLI CLI$K_STRING, DEVICE,   ILLEGAL,<'VICE'>          ; DEVICE
                        06B3  1282
                        06B3  1283 122$:   $DS_CLI CLI$K_SLASH,  NOTNUF,   140$                      ; /
                        06B7  1284
                        06B7  1285         $DS_CLI CLI$K_STRING,  0,       135$,   <'ADAPTER'>       ; / ADAPTER
                        06C3  1286         $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                   ; / ADAPTER :
                        06C7  1287         $DS_CLI CLI$K_BR,     BEGINADAPTER,125$                   ;
                        06CB  1288
                        06CB  1289 125$:   $DS_CLI CLI$K_ALNUM,  ADAPTER,  ILLEGAL                   ; Required name
                        06CF  1290         $DS_CLI <^A":">,       0,       130$                    ; Optional :
                        06D3  1291
                        06D3  1292 130$:   $DS_CLI CLI$K_BR,     ENUF,     122$                    ; Any more quals?
                        06D7  1293
                        06D7  1294 135$:   $DS_CLI CLI$K_STRING, BRIEF,    BAD_QUALIFIER,<'BRIEF'> ; DEVICE / BRIEF
                        06E1  1295         $DS_CLI CLI$K_BR,     ENUF,     122$                    ; Any more quals?
                        06E5  1296
                        06E5  1297 140$:   $DS_CLI CLI$K_SPACE,   0,       EOL                     ; Optional <SP>'s
                        06E9  1298         $DS_CLI CLI$K_BR,     OPTIONAL, GET_DEV_NAME            ; Get optional names
```

```
                        06ED  1300 ;
                        06ED  1301 ; SHOW EVENT FLAGS
                        06ED  1302 ; SHOW EVENT
                        06ED  1303 ;
                        06ED  1304
                        06ED  1305 150$:   $DS_CLI CLI$K_STRING, EVENT,     160$,   <'EVENT'>        ; EVENT
                        06F7  1306         $DS_CLI CLI$K_SPACE,  ENUF,      EOL                      ; Required <SP>'s
                        06FB  1307         $DS_CLI CLI$K_STRING, ENUF,      EOL,    <'FLAGS'>        ; EVENT FLAGS
                        0705  1308         $DS_CLI CLI$K_BR,     ENUF,      EOL
                        0709  1309
                        0709  1310 ;
                        0709  1311 ; SHOW FLAGS
                        0709  1312 ;
                        0709  1313
                        0709  1314 160$:   $DS_CLI CLI$K_STRING, FLAGS,     170$,   <'FLAGS'>        ; FLAGS
                        0713  1315         $DS_CLI CLI$K_BR,     ENUF,      EOL
                        0717  1316
                        0717  1317 ;
                        0717  1318 ; SHOW LOAD
                        0717  1319 ;
                        0717  1320
                        0717  1321 170$:   $DS_CLI CLI$K_STRING, SHOWLOAD,  180$,   <'LOAD'>         ; LOAD
                        0720  1322         $DS_CLI CLI$K_BR,     ENUF,      EOL
                        0724  1323
                        0724  1324 ;
                        0724  1325 ; SHOW MEMORY
                        0724  1326 ;
                        0724  1327
                        0724  1328 180$:   $DS_Cli <^A'M'>,        0,         190$                     ; 'M'?           [62]
                        0728  1329         $DS_CLI CLI$K_STRING, SHOWMEM,   189$,   <'EMORY'>        ; MEMORY         [62]
                        0732  1330 185$:   $DS_CLI CLI$K_SLASH,  NOTNUF,    EOL
                        0736  1331         $DS_CLI CLI$K_STRING, SHOMEMMAP,186$,   <'MAP'>
                        073E  1332         $DS_CLI CLI$K_BR,     ENUF,      185$
                        0742  1333 186$:   $DS_CLI CLI$K_STRING, SHOMEMBUF,187$,   <'BUFFER'>
                        074D  1334         $DS_CLI CLI$K_BR,     ENUF,      185$
                        0751  1335 187$:   $DS_CLI CLI$K_STRING, SHOMEMDAT,188$,   <'DATA_STRUCTURE'>
                        0764  1336         $DS_CLI CLI$K_BR,     ENUF,      185$
                        0768  1337 188$:   $DS_CLI CLI$K_STRING, SHOMEMALL,BAD_QUALIFIER, <'ALL'>
                        0770  1338         $DS_CLI CLI$K_BR,     ENUF,      185$
                        0774  1339
                        0774  1340 ;
                        0774  1341 ; SHOW MM
                        0774  1342 ;
                        0774  1343
                        0774  1344 189$:   $DS_CLI CLI$K_STRING, SHOWMM,   ILLEGAL, <'M'>            ; MM             [62]
                        077A  1345         $DS_CLI CLI$K_BR,     ENUF,      EOL
```

```
                        077E  1347
                        077E  1348 ;
                        077E  1349 ; SHOW PAGE
                        077E  1350 ;
                        077E  1351
                        077E  1352 190$:    $DS_CLI CLI$K_STRING,   NOTNUF, 195$,   <'PAGE'>         ; PAGE           [71]
                        0787  1353          $DS_CLI CLI$K_EOL,      PAGE,   ILLEGAL                  ;                [71]
                        078B  1354          $DS_CLI CLI$K_BR        ENUF,   EOL                      ;                [71]
                        078F  1355
                        078F  1356 195$:    $DS_CLI <^A'Q'>         NOTNUF, 250$                     ; Q              [42]
                        0793  1357          $DS_CLI <^A'A'>         NOTNUF, ILLEGAL                  ; QA             [42]
                        0797  1358 ;
                        0797  1359 ; SHOW QACHKLOOPLOOPS
                        0797  1360 ;
                        C797  1361
                        0797  1362          $DS_CLI CLI$K_STRING  QACL,     200$,   <'CKLOOPLOOPS'> ; QACKLOOPLOOPS [42]
                        07A7  1363          $DS_CLI CLI$K_BR      ENUF,     EOL                     ;                [42]
                        07AB  1364
                        07AB  1365 ;
                        07AB  1366 ; SHOW QADEFAULTS
                        07AB  1367 ;
                        07AB  1368
                        07AB  1369 200$:    $DS_CLI CLI$K_STRING  QADEF,    210$,   <'DEFAULTS'>    ; QADEFAULTS     [42]
                        07B8  1370          $DS_CLI CLI$K_BR      ENUF,     EOL                     ;                [42]
                        07BC  1371
                        07BC  1372 ;
                        07BC  1373 ; SHOW QAERRORPRINTS
                        07BC  1374 ;
                        07BC  1375
                        07BC  1376 210$:    $DS_CLI CLI$K_STRING  QAEP,     220$,   <'ERRORPRINTS'> ; QAERRORPRINTS [42]
                        07CC  1377          $DS_CLI CLI$K_BR      ENUF,     EOL                     ;                [42]
                        07D0  1378
                        07D0  1379 ;
                        07D0  1380 ; SHOW QAMULTIPLEPASS
                        07D0  1381 ;
                        07D0  1382
                        07D0  1383 220$:    $DS_CLI CLI$K_STRING  QAMP,     230$,   <'MULTIPLEPASS'>; QAMULTIPLEPASS[42]
                        07E1  1384          $DS_CLI CLI$K_BR      ENUF,     EOL                     ;                [42]
                        07E5  1385
                        07E5  1386 ;
                        07E5  1387 ; SHOW QASUBTESTLOOPS
                        07E5  1388 ;
                        07E5  1389
                        07E5  1390 230$:    $DS_CLI CLI$K_STRING  QASL,     240$,   <'SUBTESTLOOPS'>; QASUBTESTLOOPS[42]
                        07F6  1391          $DS_CLI CLI$K_BR      ENUF,     EOL                     ;                [42]
                        07FA  1392
                        07FA  1393 ;
                        07FA  1394 ; SHOW QATESTLOOPS
                        07FA  1395 ;
                        07FA  1396
                        07FA  1397 240$:    $DS_CLI CLI$K_STRING  QATL,     ILLEGAL,<'TESTLOOPS'>   ; QATESTLOOPS    [42]
                        0808  1398          $DS_CLI CLI$K_BR      ENUF,     EOL                     ;                [42]
```

```
080C  1400
080C  1401 250$:   $DS_CLI <^A''S''>,      NOTNUF,   300$                    ; S
0810  1402
0810  1403 ;
0810  1404 ; Minimums for SHOW S... commands:
0810  1405 ;
0810  1406 ;       SE  => SELECTED
0810  1407 ;       SEC => SECTIONS
0810  1408 ;       ST  => STATUS
0810  1409 ;       SU  => SUPPORT
0810  1410 ;
0810  1411
0810  1412         $DS_CLI <^A''E''>,      NOTNUF,   280$                    ; SE
0814  1413         $DS_CLI <^A''C''>,      ENUF,     255$                    ; SEC
0818  1414
0818  1415 ;
0818  1416 ; SHOW SECTIONS
0818  1417 ;
0818  1418
0818  1419         $DS_CLI CLI$K_BR,       BACKUP,   252$                    ; Backup over C
081C  1420
081C  1421 252$:   $DS_CLI CLI$K_STRING, SHOWSECTIONS,255$,<'CTIONS'>       ; SECTIONS
0827  1422         $DS_CLI CLI$K_BR,       ENUF,     EOL
082B  1423
082B  1424 ;
082B  1425 ; SHOW SELECTED
082B  1426 ; SHOW SELECTED/BRIEF
082B  1427 ; Spaces may appear on either side of the "/".
```

```
082B  1429 ;
082B  1430
082B  1431 255$:   $DS_CLI CLI$K_BR,     BACKUP,    257$                        ; Backup over E
082F  1432
082F  1433 257$:   $DS_CLI CLI$K_STRING, SHOWSEL,   ILLEGAL,<'ELECTED'>        ; SELECTED      [39]
083B  1434         $DS_CLI CLI$K_SLASH,  NOTNUF,    260$                        ; /
083F  1435
083F  1436         $DS_CLI CLI$K_STRING, BRIEF,     BAD_QUALIFIER,<'BRIEF'> ; SELECTED /BRIEF
0849  1437         $DS_CLI CLI$K_BR,     ENUF,      EOL
084D  1438
084D  1439 260$:   $DS_CLI CLI$K_BR,     ENUF,      EOL                         ; No /, must be EOL
0851  1440
```

```
                     0851  1442
                     0851  1443 ;
                     0851  1444 ; SHOW STATUS
                     0851  1445 ;
                     0851  1446
                     0851  1447 280$:   $DS_CLI CLI$K_STRING, SHOWSTATUS,290$,  <'TATUS'>       ; STATUS          [39]
                     085B  1448         $DS_CLI CLI$K_BR,     ENUF,      EOL                     ;                 [39]
                     085F  1449                                                                   ;                 [42]
                     085F  1450 ;
                     085F  1451 ; SHOW SUPPORT
                     085F  1452 ;
                     085F  1453
                     085F  1454 290$:   $DS_CLI CLI$K_STRING, SHOWSUP,  ILLEGAL,<'UPPORT'>       ; SUPPORT
                     086A  1455         $DS_CLI CLI$K_BR,     ENUF,      EOL
                     C86E  1456 ;
                     086E  1457 ; SHOW WIDTH
                     086E  1458 ;
                     086E  1459
                     086E  1460 300$:   $DS_CLI CLI$K_STRING, WIDTH,    310$,   <'WIDTH'>       ; WIDTH           [38]
                     0878  1461         $DS_CLI CLI$K_BR,     ENUF,      EOL                     ;                 [38]
```

```
                        ;
                        ; SHOW/BRIEF DEVICE                          dev_name_1,dev_name_2,...
                        ; SHOW/ADAPTER:name DEVICE                   dev_name_1,dev_name_2,...
                        ; SHOW ADAPTER:name/BRIEF DEVICE             dev_name_1,dev_name_2,...
                        ; SHOW/BRIEF SELECTED
                        ;
                        ; Spaces may appear on either side of the "/".
                        ; Note: if the /ADAPTER:name qualifier appears more than once
                        ; on the command line, the LAST name given is the one used.
                        ; Note: if the character is not a slash, then the current
                        ; character is invalid in the SHOW context. Thus, it is an error.
                        ; Note: this allows the /ADAPTER:name qualifier on the SHOW SELECTED
                        ; command, even though the adapter name is not used by the SELECT routine
                        ; (this is because the look-ahead is too much for this parser to handle
                        ; in that case; i.e., the parse would have to look ahead to see if this
                        ; is show selected or a show device command, and then if show selected,
                        ; issue an error).
                        ;
087C 1482 310$:   $DS_CLI CLI$K_SLASH,  NOTNUF,   ILLEGAL                  ; /
0880 1483
0880 1484 ;
0880 1485 ; Get the qualifiers: /ADAPTOR or /BRIEF.
0880 1486 ;
0880 1487
0880 1488 312$:   $DS_CLI CLI$K_STRING,   0,      318$,   <'ADAPTER'>     ; / ADAPTER
088C 1489         $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                 ; / ADAPTER :
0890 1490         $DS_CLI CLI$K_BR,     BEGINADAPTER,314$                 ;
0894 1491
0894 1492 314$:   $DS_CLI CLI$K_ALNUM,  ADAPTER,  ILLEGAL                 ; Required name
0898 1493         $DS_CLI <^A":">,        0,      316$                    ; Optional :
089C 1494
089C 1495 316$:   $DS_CLI CLI$K_BR,       0,      320$                    ; Get next qual.
08A0 1496
08A0 1497 318$:   $DS_CLI CLI$K_STRING, BRIEF,    BAD_QUALIFIER,<'BRIEF'> ; / BRIEF
08AA 1498
08AA 1499 320$:   $DS_CLI CLI$K_SLASH,  NOTNUF,   330$                    ; If not /, branch
08AE 1500         $DS_CLI CLI$K_BR,       0,      312$                    ; Get next qualifier
08B2 1501
08B2 1502 ;
08B2 1503 ; Get the keyword: either DEVICE or SELECTED.
08B2 1504 ;
08B2 1505
08B2 1506 330$:   $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL                  ; If no <SP>'s, bran
08B6 1507
08B6 1508         $DS_CLI CLI$K_STRING, DEVICE,   340$,   <'DEVICE'>      ; / ?? DEVICE
08C1 1509         $DS_CLI CLI$K_BR,     OPTIONAL, GET_DEV_NAME            ; Optional names
08C5 1510
08C5 1511 340$:   $DS_CLI CLI$K_STRING, SHOWSEL,  ILLEGAL,<'SELECTED'>   ; / ?? SELECTED [39]
08D2 1512         $DS_CLI CLI$K_BR,     ENUF,     EOL
08D6 1513
08D6 1514 ;
08D6 1515 ; End of SHOW.
08D6 1516 ;
```

```
                        .SUBTITLE       Clear Parameters and Qualifiers
                ;
                ; CLEAR parameters.
                ;
                ;       CLEAR BREAKPOINTS ALL
                ;       CLEAR BREAKPOINTS address                                       [67]
                ;       CLEAR CRD/TRACE
                ;       CLEAR ENFORCE
                ;       CLEAR EVENT ALL
                ;       CLEAR EVENT FLAGS ALL
                ;       CLEAR EVENT FLAGS x,x,...
                ;       CLEAR EVENT x,x,...
                ;       CLEAR FLAGS y,y,y,...
                ;       CLEAR y,y,y,...
                ;
CLEAR_PARAMS:
                        $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s

                ;
                ; CLEAR BREAKPOINTS ALL
                ; CLEAR BREAKPOINTS address
                ; Note: the address could start with "A". Thus, the check and backup below.
                ;

                        $DS_CLI <^A'B'>,        NOTNUF,  116$                   ; B             [65]

                        $DS_CLI CLI$K_STRING, BREAK,    115$,   < REAKPOINTS'>  ; BREAKPOINTS
                        $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s

                        $DS_CLI <^A'A'>,        NOTNUF,  110$                   ; A
                        $DS_CLI CLI$K_STRING, ALL,      100$,   <'LL'>          ; ALL
                        $DS_CLI CLI$K_BR,       ENUF,    EOL

100$:           $DS_CLI CLI$K_BR,       BACKUP,  110$                   ; Backup over A

110$:           $DS_CLI CLI$K_NUM,      ADDRESS, ILLEGAL                ; Address
                $DS_CLI CLI$K_BR,       ENUF,    EOL

115$:           $DS_CLI CLI$K_BR,       BACKUP,  180$                   ; Backup over 'B'
                                                                        ; Go check BELL

                ;+
                ; CLEAR CRD/TRACE                                                       [67]
                ;       Allow CLEAR CRD to mean CLEAR CRD/TRACE for now - until more options    [67]
                ;       are allowed. The command will eventually be like this:          [67]
                ;               CLEAR CRD/TRACE=(VERBOSE,xxxx,...)/...                  [67]
                ;-

116$:           $DS_CLI CLI$K_STRING, 0,        120$,   <'CRD'>         ; CRD           [69]
                $DS_Cli CLI$K_Slash,  NotNuf,   117$,                   ; CRD/          [69]
                $DS_Cli CLI$K_String, 0,        118$,   <'TRACE'>       ; CRD/TRACE     [69]

117$:           $DS_CLI CLI$K_Eol,      Clear_CRD_Trace,  Illegal       ; Must end EOL  [69]
                $DS_Cli CLI$K_Br,       Enuf,     Eol                   ;               [69]

                ;+
```

```
                        ;
092E  1575 ; Now check for CLEAR CRD/DEBUG/TRACE. Note that CLEAR CRD/TRACE/DEBUG doesn't  [69]
092E  1576 ; work. Note that we have seen CLEAR CRD/ so far.                               [69]
092E  1577 ;-
092E  1578
092E  1579 118$:    $DS_Cli CLI$K_String, NotNuf,   Bad_Qualifier,<'DEBUG'> ; CRD/DEBUG     [69]
0938  1580          $DS_Cli CLI$K_Slash,  NotNuf,   Bad_Qualifier,           ; CRD/DEBUG/    [69]
093C  1581          $DS_Cli CLI$K_String, 0,        Bad_Qualifier,<'TRACE'>; CRD/DEBUG/TRACE[69]
0946  1582          $DS_Cli CLI$K_Eol,    Clear_CRD_Debug, Illegal           ; Must end EOL  [69]
094A  1583          $DS_Cli CLI$K_Br,     Enuf,     Eol                      ;               [69]
```

```
                     094E  1585 ;
                     094E  1586 ; CLEAR EVENT FLAGS ALL
                     094E  1587 ; CLEAR EVENT FLAGS x,x,x,...
                     094E  1588 ; CLEAR EVENT ALL
                     094E  1589 ; CLEAR EVENT x,x,x,...
                     094E  1590 ; Spaces may appear on either side of the ",".
                     094E  1591 ;
                     094E  1592
                     094E  1593 120$:   $Ds_Cli <^A"E">,        0,       170$                        ; an E?
                     0952  1594         $Ds_Cli Cli$K_String,   0,       125$, <'NFORCE'>            ; ENFORCE
                     095D  1595         $Ds_Cli Cli$K_Eol, ClearEnforce,Illegal                      ; Must end at EOL
                     0961  1596         $Ds_Cli Cli$K_Br,     ENUF,     EOL                         ;
                     0965  1597 125$:   $DS_CLI CLI$K_STRING, EVENT,    Illegal,<'VENT'>            ; EVENT
                     096E  1598         $DS_CLI CLI$K_SPACE,    0,       ILLEGAL                     ; Required <SP>'s
                     0972  1599
                     0972  1600         $DS_CLI CLI$K_STRING, FLAGS,    130$,   <'FLAGS'>           ; "EVENT FLAGS"
                     097C  1601         $DS_CLI CLI$K_SPACE,    0,       ILLEGAL                     ; Required <SP>'s
                     0980  1602
                     0980  1603 130$:   $DS_CLI CLI$K_STRING, ALL,      140$,   <'ALL'>             ; "EVENT FLAGS ALL"
                     0988  1604
                     0988  1605 135$:   $DS_CLI CLI$K_BR,     ENUF,     EOL
                     098C  1606
                     098C  1607 140$:   $DS_CLI CLI$K_DEC,    EFN,      BAD_LIST                    ; "EVENT FLAGS xx"
                     0990  1608         $DS_CLI CLI$K_COMMA,  NOTNUF,   135$                        ; ,
                     0994  1609
                     0994  1610 160$:   $DS_CLI CLI$K_BR,       0,       140$
                     0998  1611
                     0998  1612 ;
                     0998  1613 ; CLEAR FLAGS x,x,x,...
                     0998  1614 ; CLEAR x,x,x,...
                     0998  1615 ; Spaces may appear on either side of the ",".
                     0998  1616 ;
                     0998  1617
                     0998  1618 170$:   $DS_CLI CLI$K_STRING, FLAGS,    180$,   <'FLAGS'>           ; FLAGS
                     09A2  1619         $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL                     ; Required <SP>'s
                     09A6  1620         $DS_CLI CLI$K_BR,     NOTNUF,   CLEAR_FLAGS_LOOP            ; Get the flags
                     09AA  1621
                     09AA  1622 ;
                     09AA  1623 ; CLEAR ???
                     09AA  1624 ;
                     09AA  1625
                     09AA  1626 180$:   $DS_CLI CLI$K_BR,     NOTNUF,   CLEAR_FLAGS_LOOP            ; Try the flags
```

```
09AE  1628
09AE  1629 ;
09AE  1630 ; Start of CLEAR_FLAGS loop. This command accepts a list of flags, separated
09AE  1631 ; by commas. The commas can have spaces before and after them. The flag ALL
09AE  1632 ; can be included anywhere in the list. Must not accept a CLEAR B, since this
09AE  1633 ; could be CLEAR BREAKPOINTS, CLEAR BELL, or CLEAR BINARY.
09AE  1634 ;
09AE  1635
09AE  1636 CLEAR_FLAGS_LOOP:
09AE  1637
09AE  1638          $DS_CLI CLI$K_STRING, ALL,      100$,   <'ALL'>          ; ALL
09B6  1639          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
09BA  1640
09BA  1641 100$:    $DS_CLI <^A'B'>,        NOTNUF, 120$                     ; B
09BE  1642
09BE  1643          $DS_CLI CLI$K_STRING, BELL,     110$,   <'ELL'>          ; BELL
09C6  1644          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
09CA  1645
09CA  1646 110$:    $DS_CLI CLI$K_STRING, BINARY,   BAD_LIST,<'INARY'>       ; BINARY        [45]
09D4  1647          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP     ;               [45]
09D8  1648
09D8  1649 120$:    $DS_CLI CLI$K_STRING, HALT,     130$,   <'HALT'>         ; HALT
09E1  1650          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
09E5  1651
09E5  1652 130$:    $DS_CLI <^A'I'>,        NOTNUF, 170$                     ; I
09E9  1653          $DS_CLI <^A'E'>,        NOTNUF, BAD_LIST                 ; IE
09ED  1654          $DS_CLI <^A'1'>,        IE1,    140$                     ; IE1
09F1  1655          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
09F5  1656
09F5  1657 140$:    $DS_CLI <^A'2'>,        IE2,    150$                     ; IE2
09F9  1658          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
09FD  1659
09FD  1660 150$:    $DS_CLI <^A'3'>,        IE3,    160$                     ; IE3
0A01  1661          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
0A05  1662
0A05  1663 160$:    $DS_CLI <^A'S'>,        IES,    BAD_LIST                 ; IES
0A09  1664          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
0A0D  1665
0A0D  1666 170$:    $DS_CLI CLI$K_STRING, LOOP,     180$,   <'LOOP'>         ; LOOP
0A16  1667          $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
```

```
0A1A  1669
0A1A  1670 180$:   $DS_CLI CLI$K_STRING, OPER,     190$,   <'OPERATOR'>     ; OPERATOR
0A27  1671         $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
0A2B  1672
0A2B  1673 190$:   $DS_CLI CLI$K_STRING, PROMPT,   200$,   <'PROMPT'>       ; PROMPT
0A36  1674         $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
0A3A  1675
0A3A  1676 200$:   $DS_CLI CLI$K_STRING, QUICK,    210$,   <'QUICK'>        ;                    [42]
0A44  1677         $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
0A48  1678
0A48  1679 210$:   $DS_CLI CLI$K_STRING, SEARCH,   220$,   <'SEARCH'>       ; SEARCH
0A53  1680         $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP
0A57  1681
0A57  1682 220$:   $DS_CLI CLI$K_STRING, TRACE,    230$,   <'TRACE'>        ; TRACE              [42]
CA61  1683         $DS_CLI CLI$K_BR        0,      END_CLEAR_FLAGS_LOOP     ;                    [42]
0A65  1684
0A65  1685 230$:   $DS_CLI CLI$K_STRING, VERIFY,   BAD_LIST,<'VERIFY'>      ; VERIFY             [45]
0A70  1686         $DS_CLI CLI$K_BR,       0,      END_CLEAR_FLAGS_LOOP     ;                    [45]
```

```
                     0A74  1688
                     0A74  1689 ;
                     0A74  1690 ; End of the CLEAR FLAGS loop. Check for <SP>'s, comma, <SP>'s, and then go get
                     0A74  1691 ; another flag. The <SP>'s are optional. That is, accept the following:
                     0A74  1692 ;
                     0A74  1693 ;       CLEAR TRACE,BELL,ALL
                     0A74  1694 ;       CLEAR TRACE , BELL      ,                ALL
                     0A74  1695 ;
                     0A74  1696
                     0A74  1697 END_CLEAR_FLAGS_LOOP:
                     0A74  1698
                     0A74  1699         $DS_CLI CLI$K_COMMA,  NOTNUF,   100$                    ;
                     0A78  1700         $DS_CLI CLI$K_BR,       0,      CLEAR_FLAGS_LOOP        ; Get another flag
                     0A7C  1701
                     CA7C  1702 100$:   $DS_CLI CLI$K_BR,     ENUF,     EOL                     ; Enough input
                     0A80  1703
                     0A80  1704 ;
                     0A80  1705 ; End of CLEAR.
                     0A80  1706 ;
```

```
                        .SUBTITLE       Set Parameters and Qualifiers
                ;
                ; SET parameters and qualifiers.
                ;
                ;       SET BASE
                ;       SET BREAKPOINTS address
                ;       SET CRD/TRACE                                                   [67]
                ;       SET DEFAULTS
                ;       SET ENFORCE
                ;       SET EVENT FLAGS ALL
                ;       SET EVENT FLAGS x,x,x,...
                ;       SET EVENT ALL
                ;       SET EVENT x,x,x,...
                ;       SET FLAGS y,y,y,...
                ;       SET y,y,y,...
                ;       SET LOAD
                ;       SET MEMORY n                                                    [75]
                ;       SET MM ON
                ;       SET MM OFF
                ;       SET PAGE
                ;       SET QA...
                ;       SET WIDTH
                ;
                SET_PARAMS:
                        $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s
                        $DS_CLI <^A'B''>,       NOTNUF, 112$                    ; B             [65]
                ;
                ; SET BASE address
                ;
                        $DS_CLI CLI$K_STRING, BASE,     100$,   <'ASE'>         ; BASE
                        $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL                 ; Required <SP>'s
                        $DS_CLI CLI$K_NUM,    ADDRESS,  ILLEGAL                 ; BASE Address
                        $DS_CLI CLI$K_BR,     ENUF,     EOL
```

```
0A9C       ;
0A9C       ; SET BREAKPOINTS address
0A9C       ;
0A9C
0A9C  100$:     $DS_CLI CLI$K_STRING, BREAK,     110$,    <'REAKPOINTS'>   ; BREAKPOINTS
0AAB            $DS_CLI CLI$K_SPACE,  NOTNUF,    ILLEGAL                   ; Required <SP>'s
0AAF            $DS_CLI CLI$K_NUM,    ADDRESS,   ILLEGAL                   ; BREAKPOINTS Addres
0AB3            $DS_CLI CLI$K_BR,     ENUF,      EOL
0AB7
0AB7  110$:     $DS_CLI CLI$K_BR,     BACKUP,    SET_FLAGS_LOOP            ; Backup over 'B'
0ABB                                                                       ; Check for BELL
0ABB
0ABB       ;+
0ABB       ; SET CRD/TRACE                                                               [67]
0ABB       ;        Allow SET CRD to mean SET CRD/TRACE for now - until more options     [67]
0ABB       ;        are allowed. The command will eventually be like this:               [67]
0ABB       ;                SET CRD/TRACE=(VERBOSE,xxxx,...)/...                          [67]
0ABB       ;-
0ABB
0ABB  112$:     $DS_CLI CLI$K_STRING, 0,         120$,    <'CRD'>          ; SET CRD      [67]
0AC3            $DS_Cli CLI$K_Slash,  NotNuf,    113$,                     ; CRD/         [67]
0AC7            $DS_Cli CLI$K_String, 0,         114$,    <'TRACE'>        ; CRD/TRACE    [67]
0AD1
0AD1  113$:     $DS_CLI CLI$K_Eol,    Set_CRD_Trace,    Illegal            ; Must end EOL [67]
0AD5            $DS_Cli CLI$K_Br,     Enuf,      Eol                       ;              [65]
0AD9
0AD9       ;+
0AD9       ; Now check for SET CRD/DEBUG/TRACE. Note that SET CRD/TRACE/DEBUG doesn't work.[69]
0AD9       ; Note that we have seen SET CRD/ so far.                                     [69]
0AD9       ;-
0AD9
0AD9  114$:     $DS_Cli CLI$K_String, NotNuf,    Bad_Qualifier,<'DEBUG'> ; CRD/DEBUG      [67]
0AE3            $DS_Cli CLI$K_Slash,  NotNuf,    Bad_Qualifier,          ; CRD/DEBUG/     [67]
0AE7            $DS_Cli CLI$K_String, 0,         Bad_Qualifier,<'TRACE'>; CRD/DEBUG/TRACE[67]
0AF1            $DS_Cli CLI$K_Eol,    Set_CRD_Debug,    Illegal          ; Must end EOL   [67]
0AF5            $DS_Cli CLI$K_Br,     Enuf,      Eol                     ;                [65]
```

```
                    0AF9  1783 ;
                    0AF9  1784 ; SET DEFAULTS x,...
                    0AF9  1785 ;
                    0AF9  1786
                    0AF9  1787 120$:    $DS_CLI CLI$K_STRING, DEFAULT,  130$,   <'DEFAULTS'>    ; DEFAULTS
                    0B06  1788          $DS_CLI CLI$K_BR,     NOTNUF,   SET_DEFAULT_PARAMS      ; Get parameters
```

```
                     0B0A  1790 ;
                     0B0A  1791 ; SET ENFORCE
                     0B0A  1792 ; SET EVENT FLAGS ALL
                     0B0A  1793 ; SET EVENT FLAGS x,x,x,...
                     0B0A  1794 ; SET EVENT ALL
                     0B0A  1795 ; SET EVENT x,x,x,...
                     0B0A  1796 ;
                     0B0A  1797
                     0B0A  1798 130$:    $Ds_Cli <^A'E'>,          0,       180$                    ; an E?
                     0B0E  1799          $Ds_Cli Cli$K_String,     0,       135$, <'NFORCE'>        ; ENFORCE
                     0B19  1800          $Ds_Cli Cli$K_Eol,   SetEnforce,   Illegal                 ; Must end at EOL
                     0B1D  1801          $Ds_Cli Cli$K_Br,       ENUF,      EOL                     ;
                     0B21  1802
                     0B21  1803 135$:    $DS_CLI CLI$K_STRING, EVENT,     Illegal,<'VENT'>        ; EVENT
                     0B2A  1804          $DS_CLI CLI$K_SPACE,    0,       ILLEGAL                 ; Required <SP>'s
                     0B2E  1805          $DS_CLI CLI$K_STRING, FLAGS,     140$,   <'FLAGS'>       ; ''EVENT FLAGS''
                     0B38  1806          $DS_CLI CLI$K_SPACE,    0,       ILLEGAL                 ; Required <SP>'s
                     0B3C  1807
                     0B3C  1808 140$:    $DS_CLI CLI$K_STRING, ALL,       150$,   <'ALL'>         ; ''EVENT FLAGS ALL''
                     0B44  1809
                     0B44  1810 ,45$:    $DS_CLI CLI$K_BR,       ENUF,      EOL
                     0B48  1811
                     0B48  1812 150$:    $DS_CLI CLI$K_DEC,      EFN,       BAD_LIST                ; ''EVENT FLAGS xx''
                     0B4C  1813          $DS_CLI CLI$K_COMMA,  NOTNUF,      145$                    ; ,
                     0B50  1814          $DS_CLI CLI$K_BR,       0,       150$
```

```
                        0B54  1816 ;
                        0B54  1817 ; SET FLAGS
                        0B54  1818 ;
                        0B54  1819
                        0B54  1820 180$:   $DS_CLI CLI$K_STRING, FLAGS,    190$,   <'FLAGS'>       ; FLAGS
                        0B5E  1821         $DS_CLI CLI$K_SPACE,   0,       ILLEGAL                 ; <SP>'s required
                        0B62  1822
                        0B62  1823         $DS_CLI CLI$K_BR,     NOTNUF,   SET_FLAGS_LOOP                  ; Get flags
                        0B66  1824
                        0B66  1825 ;
                        0B66  1826 ; SET LOAD file-spec
                        0B66  1827 ;
                        0B66  1828
                        0B66  1829 190$:   $DS_CLI <^A'L'>,      NOTNUF,   220$                    ; L
                        0B6A  1830         $DS_CLI <^A'O'>,      NOTNUF,   ILLEGAL                 ; LO
                        0B6E  1831         $DS_CLI CLI$K_STRING, SETLOAD,  200$,   <'AD'>          ; LOAD
                        0B75  1832         $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL                 ; Required spaces
                        0B79  1833         $DS_CLI CLI$K_BR,     REQUIRED, GET_FILE_SPEC           ; Required file-spec
                        0B7D  1834
                        0B7D  1835 200$:   $DS_CLI CLI$K_BR,     BACKUP,   210$                    ; Backup over 'O'
                        0B81  1836
                        0B81  1837 210$:   $DS_CLI CLI$K_BR,     BACKUP,   SET_FLAGS_LOOP                  ; Backup ove
                        0B85  1838
                        0B85  1839 ;                                                                        [75]
                        0B85  1840 ; SET MEMORY n                                                           [75]
                        0B85  1841 ;                                                                        [75]
                        0B85  1842
                        0B85  1843 220$:   $DS_CLI <^A'M'>,      NOTNUF, 235$                      ; M             [75]
                        0B89  1844
                        0B89  1845         $DS_CLI CLI$K_STRING, SETMEM, 225$,   <'EMORY'>          ; MEMORY        [75]
                        0B93  1846         $DS_CLI CLI$K_SPACE,  0,      ILLEGAL                   ; Required <SP>'s[75
                        0B97  1847         $DS_CLI CLI$K_DEC,    DATA,   ILLEGAL                   ; data          [75]
                        0B9B  1848         $DS_CLI CLI$K_BR,     ENUF,   EOL                       ;               [75]
                        0B9F  1849
                        0B9F  1850 ;
                        0B9F  1851 ; SET MM ON
                        0B9F  1852 ; SET MM OFF
                        0B9F  1853 ;
                        0B9F  1854
                        0B9F  1855 225$:   $DS_CLI CLI$K_STRING, NOTNUF,  235$,   <'MM'>          ; MM            [75]
                        0BA6  1856         $DS_CLI CLI$K_SPACE,   0,      ILLEGAL                   ; Required <SP>'s
                        0BAA  1857         $DS_CLI <^A'O'>,      NOTNUF,  ILLEGAL                   ; MM O
                        0BAE  1858         $DS_CLI CLI$K_STRING, MMON,    230$,   <'N'>             ; MM ON
                        0BB4  1859         $DS_CLI CLI$K_BR,     ENUF,    EOL
                        0BB8  1860
                        0BB8  1861 230$:   $DS_CLI CLI$K_STRING, MMOFF,   ILLEGAL,<'FF'>            ; MM OFF
                        0BBF  1862         $DS_CLI CLI$K_BR,     ENUF,    EOL
                        0BC3  1863
                        0BC3  1864 ;
                        0BC3  1865 ; SET PAGE n
                        0BC3  1866 ;
                        0BC3  1867
                        0BC3  1868 235$:   $DS_CLI CLI$K_STRING, NOTNUF, 240$,   <'PAGE'>          ; PAGE          [71]
                        0BCC  1869         $DS_CLI CLI$K_SPACE,  NOTNUF, ILLEGAL                   ; required space[71]
                        0BD0  1870         $DS_CLI CLI$K_DEC,    DATA,   ILLEGAL                   ; data          [71]
                        0BD4  1871         $DS_CLI CLI$K_BR,     PAGE,   236$                      ;               [71]
                        0BD8  1872 236$:   $DS_CLI CLI$K_BR,     ENUF,   EOL                       ;               [71]
```

0BDC 1873

```
                        0BDC  1875 ;
                        0BDC  1876 ; SET QACHKLOOPLOOPS x
                        0BDC  1877 ; SET QADEFAULTS
                        0BDC  1878 ; SET QAERRORPRINTS  x
                        0BDC  1879 ; SET QAMULTIPLEPASS x
                        0BDC  1880 ; SET QASUBTESTLOOPS x
                        0BDC  1881 ; SET QATESTLOOPS    x
                        0BDC  1882 ;
                        0BDC  1883
                        0BDC  1884 240$:   $DS_CLI <^A''Q''>       NOTNUF,   300$                          ; Q              [42]
                        0BE0  1885         $DS_CLI <^A''A''>       NOTNUF,   242$                          ; QA             [42]
                        0BE4  1886         $DS_CLI CLI$K_BR,         0,      245$                          ; Skip over next one
                        0BE8  1887
                        0BE8  1888 242$:   $DS_CLI CLI$K_BR,       BACKUP,   SET_FLAGS_LOOP                ; Backup over ''Q'' -
                                                                                                           ; might be SET QUICK
                        0BEC  1889
                        0BEC  1890
                        0BEC  1891 245$:   $DS_CLI CLI$K_STRING    QACL,     250$,   <'CKLOOPLOOPS'> ; QACKLOOPLOOPS [42]
                        0BFC  1892         $DS_CLI CLI$K_SPACE,    NOTNUF,   ILLEGAL                       ; Required <SP>'s
                        0C00  1893         $DS_CLI CLI$K_DEC,      DATA,     ILLEGAL                       ; QA... data
                        0C04  1894         $DS_CLI CLI$K_BR        ENUF,     EOL                           ;                [42]
                        0C08  1895
                        0C08  1896 250$:   $DS_CLI CLI$K_STRING    QADEF,    260$,   <'DEFAULTS'>    ; QADEFAULTS     [42]
                        0C15  1897         $DS_CLI CLI$K_BR        ENUF,     EOL                           ; No data        [42]
                        0C19  1898
                        0C19  1899 260$:   $DS_CLI CLI$K_STRING    QAEP,     270$,   <'ERRORPRINTS'> ; QAERRORPRINTS [42]
                        0C29  1900         $DS_CLI CLI$K_SPACE,    NOTNUF,   ILLEGAL                       ; Required <SP>'s
                        0C2D  1901         $DS_CLI CLI$K_DEC,      DATA,     ILLEGAL                       ; QA... data
                        0C31  1902         $DS_CLI CLI$K_BR        ENUF,     EOL                           ;                [42]
                        0C35  1903
                        0C35  1904 270$:   $DS_CLI CLI$K_STRING    QAMP,     280$,   <'MULTIPLEPASS'>; QAMULTIPLEPASS[42]
                        0C46  1905         $DS_CLI CLI$K_SPACE,    NOTNUF,   ILLEGAL                       ; Required <SP>'s
                        0C4A  1906         $DS_CLI CLI$K_DEC,      DATA,     ILLEGAL                       ; QA... data
                        0C4E  1907         $DS_CLI CLI$K_BR        ENUF,     EOL                           ;                [42]
                        0C52  1908
                        0C52  1909 280$:   $DS_CLI CLI$K_STRING    QASL,     290$,   <'SUBTESTLOOPS'>; QASUBTESTLOOPS[42]
                        0C63  1910         $DS_CLI CLI$K_SPACE,    NOTNUF,   ILLEGAL                       ; Required <SP>'s
                        0C67  1911         $DS_CLI CLI$K_DEC,      DATA,     ILLEGAL                       ; QA... data
                        0C6B  1912         $DS_CLI CLI$K_BR        ENUF,     EOL                           ;                [42]
                        0C6F  1913
                        0C6F  1914 290$:   $DS_CLI CLI$K_STRING    QATL,     ILLEGAL,<'TESTLOOPS'>   ; QATESTLOOPS    [42]
                        0C7D  1915         $DS_CLI CLI$K_SPACE,    NOTNUF,   ILLEGAL                       ; Required <SP>'s
                        0C81  1916         $DS_CLI CLI$K_DEC,      DATA,     ILLEGAL                       ; QA... data
                        0C85  1917         $DS_CLI CLI$K_BR        ENUF,     EOL                           ;                [42]
                        0C89  1918
                        0C89  1919 ;
                        0C89  1920 ; SET WIDTH x
                        0C89  1921 ;
                        0C89  1922
                        0C89  1923 300$:   $DS_CLI CLI$K_STRING, WIDTH,    SET_FLAGS_LOOP,<'WIDTH'>; WIDTH
                        0C93  1924         $DS_CLI CLI$K_SPACE,    NOTNUF,   ILLEGAL                       ; Required <SP>'s
                        0C97  1925         $DS_CLI CLI$K_DEC,      DATA,     ILLEGAL                       ; WIDTH x
                        0C9B  1926         $DS_CLI CLI$K_BR,       ENUF,     EOL
```

```
                        ;
                        ; Start of the flags loop. That is, there can be any number of these flags
                        ; for a given SET FLAGS command. The flags must be separated by commas.
                        ; Spaces may be before and after the commas. Each flag that starts with
                        ; character(s) from other SET keywords (e.g., LOOP -> LOAD, BELL ->
                        ; BASE, BINARY -> BREAKPOINTS, ...) must recognize the minimum set of
                        ; characters. That is "BE" rather than just "B". This is so that a SET B
                        ; command will be recognized as INCOMPLETE, rather that accepting the B as
                        ; B in BELL. Note: this does not apply to QUICK vs. QA... so as to remain
                        ; compatible with pre-QA Supervisors.
                        ;

                        SET_FLAGS_LOOP:

                                $DS_CLI CLI$K_STRING, ALL,      100$,   <'ALL'>         ; ALL
                                $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP

                        100$:   $DS_CLI <^A'B'>,        NOTNUF,  120$                   ; B

                                $DS_CLI CLI$K_STRING, BELL,     110$,   <'ELL'>         ; BELL
                                $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP

                        110$:   $DS_CLI CLI$K_STRING, BINARY,   BAD_LIST,<'INARY'>      ; BINARY        [45]
                                $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP      ;               [45]

                        120$:   $DS_CLI CLI$K_STRING, DEFAULT,  125$,   <'DEFAULT'>     ; DEFAULT
                                $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP      ;

                        125$:   $DS_CLI CLI$K_STRING, HALT,     130$,   <'HALT'>        ; HALT
                                $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP

                        130$:   $DS_CLI <^A'I'>,        NOTNUF,  170$                   ; I
                                $DS_CLI <^A'E'>,        NOTNUF,  BAD_LIST               ; IE
                                $DS_CLI <^A'1'>,        IE1,     140$                   ; IE1
                                $DS_CLI CLI$K_BR,       0,       END_SET_FLAGS_LOOP

                        140$:   $DS_CLI <^A'2'>,        IE2,     150$                   ; IE2
                                $DS_CLI CLI$K_BR,       0,       END_SET_FLAGS_LOOP

                        150$:   $DS_CLI <^A'3'>,        IE3,     160$                   ; IE3
                                $DS_CLI CLI$K_BR,       0,       END_SET_FLAGS_LOOP

                        160$:   $DS_CLI <^A'S'>,        IES,     BAD_LIST               ; IES
                                $DS_CLI CLI$K_BR,       0,       END_SET_FLAGS_LOOP

                        170$:   $DS_CLI <^A'L'>,        NOTNUF,  180$                   ; L
                                $DS_CLI <^A'O'>,        NOTNUF,  BAD_LIST               ; LO
                                $DS_CLI CLI$K_STRING, LOOP,      BAD_LIST,<'OP'>        ; LOOP
                                $DS_CLI CLI$K_BR,       0,       END_SET_FLAGS_LOOP
```

```
OD21  1978 180$:   $DS_CLI CLI$K_STRING, OPER,     190$,   <'OPERATOR'>     ; OPERATOR
OD2E  1979         $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP
OD32  1980
OD32  1981 190$:   $DS_CLI CLI$K_STRING, PROMPT,   200$,   <'PROMPT'>       ; PROMPT
OD3D  1982         $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP
OD41  1983
OD41  1984 200$:   $DS_CLI CLI$K_STRING, QUICK,    210$,   <'QUICK'>        ; QUICK          [42]
OD4B  1985         $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP
OD4F  1986
OD4F  1987 210$:   $DS_CLI CLI$K_STRING, SEARCH,   220$,   <'SEARCH'>       ; SEARCH
OD5A  1988         $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP
OD5E  1989
OD5E  1990 220$:   $DS_CLI CLI$K_STRING, TRACE,    230$,   <'TRACE'>        ; TRACE          [42]
OD68  1991         $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP       ;                [42]
CD6C  1992
OD6C  1993 230$:   $DS_CLI CLI$K_STRING, VERIFY,   BAD_LIST,<'VERIFY'>      ; VERIFY         [45]
OD77  1994         $DS_CLI CLI$K_BR,       0,      END_SET_FLAGS_LOOP       ;                [45]
```

```
                        ;
                        ; End of the flags loop.
                        ;

                        END_SET_FLAGS_LOOP:

                                $DS_CLI CLI$K_COMMA,  NOTNUF,   100$                    ;
                                $DS_CLI CLI$K_BR,       0,      SET_FLAGS_LOOP          ; Get next flag

                        100$:   $DS_CLI CLI$K_BR,     ENUF,     EOL                     ; Enough input

                        ;
                        ; End of SET.
                        ;
```

```
                     0D87  2011          .SUBTITLE       Examine Qualifiers and Parameters
                     0D87  2012 ;
                     0D87  2013 ; EXAMINE qualifiers and parameter.
                     0D87  2014 ; This allows the qualifiers to appear either before the address value, or
                     0D87  2015 ; after. Thus, the following are legal:
                     0D87  2016 ;
                     0D87  2017 ;       EXAMINE/BYTE 200
                     0D87  2018 ;       EXAMINE 200/BYTE
                     0D87  2019 ;       EXAMINE/DECIMAL/BYTE 200
                     0D87  2020 ;       EXAMINE/DECIMAL 200/BYTE
                     0D87  2021 ;       EXAMINE 200/DECIMAL/BYTE
                     0D87  2022 ;
                     0D87  2023
                     0D87  2024 EXAMINE_PARAMS:
                     0D87  2025          $DS_CLI CLI$K_SLASH,  NOTNUF,   210$                    ; /
```

```
                        0D8B  2027 ;
                        0D8B  2028 ; EXAMINE/ASCII
                        0D8B  2029 ;
                        0D8B  2030
                        0D8B  2031          $DS_CLI CLI$K_STRING, ASCII,    120$,   <'ASCII'>       ; /ASCII
                        0D95  2032          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0D99  2033
                        0D99  2034 ;
                        0D99  2035 ; EXAMINE/BYTE
                        0D99  2036 ;
                        0D99  2037
                        0D99  2038 120$:    $DS_CLI CLI$K_STRING, BYTE,     130$,   <'BYTE'>        ; /BYTE
                        0DA2  2039          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0DA6  2040
                        CDA6  2041 ;
                        0DA6  2042 ; EXAMINE/DECIMAL
                        0DA6  2043 ;
                        0DA6  2044
                        0DA6  2045 130$:    $DS_CLI CLI$K_STRING, DEC,      140$,   <'DECIMAL'>     ; /DECIMAL
                        0DB2  2046          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0DB6  2047
                        0DB6  2048 ;
                        0DB6  2049 ; EXMAINE/HEXADECIMAL
                        0DB6  2050 ;
                        0DB6  2051
                        0DB6  2052 140$:    $DS_CLI CLI$K_STRING, HEX,      150$,   <'HEXADECIMAL'> ; /HEXADECIMAL
                        0DC6  2053          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0DCA  2054
                        0DCA  2055 ;
                        0DCA  2056 ; EXAMINE/LONGWORD
                        0DCA  2057 ;
                        0DCA  2058
                        0DCA  2059 150$:    $DS_CLI CLI$K_STRING, LONG,     160$,   <'LONGWORD'>    ; /LONGWORD
                        0DD7  2060          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0DDB  2061
                        0DDB  2062 ;
                        0DDB  2063 ; EXAMINE/NEXT:n
                        0DDB  2064 ;
                        0DDB  2065
                        0DDB  2066 160$:    $DS_CLI CLI$K_STRING,   0,      190$,   <'NEXT'>        ; /NEXT
                        0DE4  2067          $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                 ; : or =
                        0DE8  2068          $DS_CLI CLI$K_DEC,    NEXT,     ILLEGAL                 ; : data
                        0DEC  2069          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0DF0  2070
                        0DF0  2071 ;
                        0DF0  2072 ; EXAMINE/OCTAL
                        0DF0  2073 ;
                        0DF0  2074
                        0DF0  2075 190$:    $DS_CLI CLI$K_STRING, OCT,      200$,   <'OCTAL'>       ; /OCTAL
                        0DFA  2076          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
                        0DFE  2077
                        0DFE  2078 ;
                        0DFE  2079 ; EXAMINE/WORD
                        0DFE  2080 ;
                        0DFE  2081
                        0DFE  2082 200$:    $DS_CLI CLI$K_STRING, WORD,     BAD_QUALIFIER,<'WORD'>  ; /WORD
                        0E07  2083          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS
```

```
                                0E0B  2085 ;
                                0E0B  2086 ; Get register number or address. If have already gotten one of these,
                                0E0B  2087 ; then assume we are at the end of the line. Note: spaces must always
                                0E0B  2088 ; appear before the register or address.
                                0E0B  2089 ;
                                0E0B  2090
                                0E0B  2091 210$:   $DS_CLI CLI$K_BIF,    IF_REG_OR_ADR,310$                ; If already gotten,
                                0E0F  2092                                                                   ;   assume EOL.
                                0E0F  2093
                                0E0F  2094         $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL                 ; Required <SP>'s
                                0E13  2095
```

```
                        ;
                        ; EXAMINE AP
                        ; This could be AP, or it could be Axxx = address.
                        ;

                                $DS_CLI <^A'A'>,      REG12,    230$          ; A
                                $DS_CLI <^A'P'>,      REGP,     220$          ; AP
                                $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS ; Go back for more

220$:                           $DS_CLI CLI$K_BR,     BACKUP,   300$          ; Backup over "A"

                        ;
                        ; EXAMINE FP
                        ; This could be FP, or it could be Fxxx = address.
                        ;

230$:                           $DS_CLI <^A'F'>,      REG13,    250$          ; F
                                $DS_CLI <^A'P'>,      REGP,     240$          ; FP
                                $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS ; Go back for more

240$:                           $DS_CLI CLI$K_BR,     BACKUP,   300$          ; Backup over "F"
```

```
                     ;
                     ; EXAMINE PC
                     ;

0E33  2123 250$:   $DS_CLI <^A'P'>,        0,      280$                    ; P
0E37  2124         $DS_CLI <^A'C'>,      REG15,   260$                    ; PC
0E3B  2125         $DS_CLI CLI$K_BR,       0,     EXAMINE_PARAMS          ; Go back for more
0E3F  2126
                     ;
                     ; EXAMINE PSL
                     ;

0E3F  2131 260$:   $DS_CLI CLI$K_STRING, REG16,   270$,   <'SL'>          ; PSL
0E46  2132         $DS_CLI CLI$K_BR,       0,     EXAMINE_PARAMS          ; Go back for more
                     ;
                     ; EXAMINE Pxx processor register
                     ;

0E4A  2138 270$:   $DS_CLI CLI$K_NUM,    PREGN,   ILLEGAL                 ; Pxx
0E4E  2139         $DS_CLI CLI$K_BR       0,     EXAMINE_PARAMS          ; Go back for more
```

```
                        ;
                        ; EXAMINE Rxx general register
                        ;
                        
0E52  2145 280$:    $DS_CLI <^A'R'>,      REGN,     290$                    ; R
0E56  2146          $DS_CLI CLI$K_DEC,    ADDRESS,  ILLEGAL                 ; Rxx
0E5A  2147          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS          ; Go back for more
                        
                        ;
                        ; EXAMINE SP
                        ;
                        
0E5E  2153 290$:    $DS_CLI <^A'S'>,      REG14,    300$                    ; S
0E62  2154          $DS_CLI <^A'P'>,      REGP,     ILLEGAL                 ; SP
0E66  2155          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS          ; Go back for more
                        
                        ;
                        ; EXAMINE address
                        ;
                        
0E6A  2161 300$:    $DS_CLI CLI$K_NUM,    ADDRESS,  ILLEGAL                 ; Address is data
0E6E  2162          $DS_CLI CLI$K_BR,       0,      EXAMINE_PARAMS          ; Go back for more
                        
                        ;
                        ; End of all the examine qualifiers and parameters. Assume EOL.
                        ;
                        
0E72  2168 310$:    $DS_CLI CLI$K_BR      ENUF,     EOL                     ; Enough input
                        ;
                        ; End of EXAMINE.
                        ;
```

```
                        0E76  2173          .SUBTITLE       Deposit Parameters and Qualifiers
                        0E76  2174 ;
                        0E76  2175 ; DEPOSIT qualifiers and parameters.
                        0E76  2176 ; This allows the qualifiers to appear either after the DEPOSIT command,
                        0E76  2177 ; or after the data value. That is, the following are legal:
                        0E76  2178 ;
                        0E76  2179 ;       DEPOSIT/BYTE 3000 30
                        0E76  2180 ;       DEPOSIT 3000 30/BYTE
                        0E76  2181 ;       DEPOSIT/BYTE 3000 30/DECIMAL
                        0E76  2182 ;
                        0E76  2183
                        0E76  2184 DEPOSIT_PARAMS:
                        0E76  2185          $DS_CLI CLI$K_SLASH,  NOTNUF,   200$                    ; /
```

```
                        ;
                        ; DEPOSIT/BYTE
                        ;
                        
                                $DS_CLI CLI$K_STRING, BYTE,     120$,   <'BYTE'>        ; /BYTE
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
                        
                        ;
                        ; DEPOSIT/DECIMAL
                        ;
                        
120$:                   $DS_CLI CLI$K_STRING, DEC,      130$,   <'DECIMAL'>     ; /DECIMAL
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
                        
                        ;
                        ; DEPOSIT/HEXADECIMAL
                        ;
                        
130$:                   $DS_CLI CLI$K_STRING, HEX,      140$,   <'HEXADECIMAL'> ; /HEXADECIMAL
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
                        
                        ;
                        ; DEPOSIT/LONGWORD
                        ;
                        
140$:                   $DS_CLI CLI$K_STRING, LONG,     150$,   <'LONGWORD'>    ; /LONGWORD
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
                        
                        ;
                        ; DEPOSIT/NEXT:n
                        ;
                        
150$:                   $DS_CLI CLI$K_STRING,   0,      180$,   <'NEXT'>        ; /NEXT
                                $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                 ; : or =
                                $DS_CLI CLI$K_DEC,    NEXT,     ILLEGAL                 ; : data
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
                        
                        ;
                        ; DEPOSIT/OCTAL
                        ;
                        
180$:                   $DS_CLI CLI$K_STRING, OCT,      190$,   <'OCTAL'>       ; /OCTAL
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
                        
                        ;
                        ; DEPOSIT/WORD
                        ;
                        
190$:                   $DS_CLI CLI$K_STRING, WORD,     BAD_QUALIFIER,<'WORD'>  ; /WORD
                                $DS_CLI CLI$K_BR,       0,      DEPOSIT_PARAMS
```

```
                        0EEC  2238
                        0EEC  2239 ;
                        0EEC  2240 ; Get register number or address. If have already gotten one of these,
                        0EEC  2241 ; then assume we are at the end of the line. Note: spaces must always appear
                        0EEC  2242 ; before the register name or address.
                        0EEC  2243 ;
                        0EEC  2244
                        0EEC  2245 200$:   $DS_CLI CLI$K_BIF,    IF_REG_OR_ADR,310$          ; If already gotten,
                        0EF0  2246                                                             ;   assume EOL.
                        0EF0  2247
                        0EF0  2248         $DS_CLI CLI$K_SPACE,  NOTNUF,   ILLEGAL           ; Required <SP>'s
                        0EF4  2249
                        0EF4  2250 ;
                        0EF4  2251 ; DEPOSIT AP
                        0EF4  2252 ; This could be AP, or it could be Axxx = address.
                        0EF4  2253 ;
                        0EF4  2254
                        0EF4  2255         $DS_CLI <^A'A'>,      REG12,    220$            ; A
                        0EF8  2256         $DS_CLI <^A'P'>,      REGP,     210$            ; AP
                        0EFC  2257         $DS_CLI CLI$K_BR,       0,      300$            ; Get data value
                        0F00  2258
                        0F00  2259 210$:   $DS_CLI CLI$K_BR,     BACKUP,   290$            ; Backup over 'A'
                        0F04  2260
                        0F04  2261 ;
                        0F04  2262 ; DEPOSIT FP
                        0F04  2263 ; This could be FP, or it could be Fxxx = address.
                        0F04  2264 ;
                        0F04  2265
                        0F04  2266 220$:   $DS_CLI <^A'F'>,      REG13,    240$            ; F
                        0F08  2267         $DS_CLI <^A'P'>,      REGP,     230$            ; FP
                        0F0C  2268         $DS_CLI CLI$K_BR,       0,      300$            ; Get data value
                        0F10  2269
                        0F10  2270 230$:   $DS_CLI CLI$K_BR,     BACKUP,   290$            , Backup over 'F'
```

```
                        0F14  2272
                        0F14  2273 ;
                        0F14  2274 ; DEPOSIT PC
                        0F14  2275 ;
                        0F14  2276
                        0F14  2277 240$:    $DS_CLI <^A'P'>,        0,      270$                    ; P
                        0F18  2278          $DS_CLI <^A'C'>,      REG15,    250$                    ; PC
                        0F1C  2279          $DS_CLI CLI$K_BR,       0,      300$                    ; Get data value
                        0F20  2280
                        0F20  2281 ;
                        0F20  2282 ; DEPOSIT PSL
                        0F20  2283 ;
                        0F20  2284
                        0F20  2285 250$:    $DS_CLI CLI$K_STRING, REG16,    260$,   <'SL'>          ; PSL
                        0F27  2286          $DS_CLI CLI$K_BR,       0,      300$                    ; Get data value
                        0F2B  2287
                        0F2B  2288 ;
                        0F2B  2289 ; DEPOSIT Pxx processor register
                        0F2B  2290 ;
                        0F2B  2291
                        0F2B  2292 260$:    $DS_CLI CLI$K_NUM,    PREGN,    ILLEGAL                 ; Pxx
                        0F2F  2293          $DS_CLI CLI$K_BR        0,      300$                    ; Get data value
```

```
0F33  2295
0F33  2296 ;
0F33  2297 ; DEPOSIT Rxx general register
0F33  2298 ;
0F33  2299
0F33  2300 270$:   $DS_CLI <^A"R">,      REGN,      280$                    ; R
0F37  2301         $DS_CLI CLI$K_DEC,    ADDRESS,   ILLEGAL                 ; Rxx
0F3B  2302         $DS_CLI CLI$K_BR,       0,       300$                    ; Get data value
0F3F  2303
0F3F  2304 ;
0F3F  2305 ; DEPOSIT SP
0F3F  2306 ;
0F3F  2307
0F3F  2308 280$:   $DS_CLI <^A"S">,      REG14,     290$                    ; S
0F43  2309         $DS_CLI <^A"P">,      REGP,      ILLEGAL                 ; SP
0F47  2310         $DS_CLI CLI$K_BR,       0,       300$                    ; Get data value
0F4B  2311
0F4B  2312 ;
0F4B  2313 ; DEPOSIT address
0F4B  2314 ;
0F4B  2315
0F4B  2316 290$:   $DS_CLI CLI$K_NUM,    ADDRESS,   ILLEGAL                 ; Address value
0F4F  2317
0F4F  2318 ;
0F4F  2319 ; DEPOSIT address data
0F4F  2320 ;
0F4F  2321
0F4F  2322 300$:   $DS_CLI CLI$K_SPACE,    0,       ILLEGAL                 ; Required <SP>'s
0F53  2323         $DS_CLI CLI$K_NUM,    DATA,      ILLEGAL                 ; Data value
0F57  2324         $DS_CLI CLI$K_BR,       0,       DEPOSIT_PARAMS          ; Go back for more
0F5B  2325                                                                 ;   qualifiers.
0F5B  2326 ;
0F5B  2327 ; End of all the DEPOSIT qualifiers and parameters. Assume EOL.
0F5B  2328 ;
0F5B  2329
0F5B  2330 310$:   $DS_CLI CLI$K_BR,     ENUF,      EOL                     ; Enough input
0F5F  2331 ;
0F5F  2332 ; End of DEPOSIT.
0F5F  2333 ;
```

```
                                0F5F  2335          .SUBTITLE       Set Default Parameters
                                0F5F  2336 ;
                                0F5F  2337 ; SET DEFAULT parameters.
                                0F5F  2338 ;
                                0F5F  2339 ; This allows a list of sizes and radixes, each one separated
                                0F5F  2340 ; by a comma. Spaces may appear on either side of the commas.
                                0F5F  2341 ; If conflicting radixes (sizes) are given, the "biggest" one
                                0F5F  2342 ; is used. That is,
                                0F5F  2343 ;       HEXADECIMAL > DECIMAL > OCTAL
                                0F5F  2344 ;       LONGWORD > WORD > BYTE
                                0F5F  2345 ;
                                0F5F  2346
                                0F5F  2347 SET_DEFAULT_PARAMS:
                                0F5F  2348          $DS_CLI CLI$K_SPACE     0,      ILLEGAL                 ; Required <SP>'s
```

```
                        0F63  2350 ;
                        0F63  2351 ; SET DEFAULT BYTE
                        0F63  2352 ;
                        0F63  2353
                        0F63  2354 100$:   $DS_CLI CLI$K_STRING, BYTE,      110$,   <'BYTE'>         ; BYTE
                        0F6C  2355         $DS_CLI CLI$K_BR,       0,       160$
                        0F70  2356
                        0F70  2357 ;
                        0F70  2358 ;  SET DEFAULT DECIMAL
                        0F70  2359 ;
                        0F70  2360
                        0F70  2361 110$:   $DS_CLI CLI$K_STRING, DEC,       120$,   <'DECIMAL'>      ; DECIMAL
                        0F7C  2362         $DS_CLI CLI$K_BR,       0,       160$
                        0F80  2363
                        CF80  2364 ;
                        0F80  2365 ; SET DEFAULT HEXADECIMAL
                        0F80  2366 ;
                        0F80  2367
                        0F80  2368 120$:   $DS_CLI CLI$K_STRING, HEX,       130$,   <'HEXADECIMAL'> ; HEXADECIMAL
                        0F90  2369         $DS_CLI CLI$K_BR,       0,       160$
                        0F94  2370
                        0F94  2371 ;
                        0F94  2372 ; SET DEFAULT LONGWORD
                        0F94  2373 ;
                        0F94  2374
                        0F94  2375 130$:   $DS_CLI CLI$K_STRING, LONG,      140$,   <'LONGWORD'>     ; LONGWORD
                        0FA1  2376         $DS_CLI CLI$K_BR,       0,       160$
                        0FA5  2377
                        0FA5  2378 ;
                        0FA5  2379 ; SET DEFAULT OCTAL
                        0FA5  2380 ;
                        0FA5  2381
                        0FA5  2382 140$:   $DS_CLI CLI$K_STRING, OCT,       150$,   <'OCTAL'>        ; OCTAL
                        0FAF  2383         $DS_CLI CLI$K_BR,       0,       160$
                        0FB3  2384
                        0FB3  2385 ;
                        0FB3  2386 ; SET DEFAULT WORD
                        0FB3  2387 ;
                        0FB3  2388
                        0FB3  2389 150$:   $DS_CLI CLI$K_STRING, WORD,      BAD_LIST,<'WORD'>        ; WORD
                        0FBC  2390         $DS_CLI CLI$K_BR,       0,       160$
```

```
                     2392 ;
                     2393 ; If there is a comma, get it and go back for more.
                     2394 ; If not, it must be end of line time.
                     2395 ;
                     2396
OFC0                 2397 160$:   $DS_CLI CLI$K_COMMA,  NOTNUF,   170$          ; ,
OFC4                 2398         $DS_CLI CLI$K_BR,       0,      100$          ; Go back for more
OFC8                 2399
OFC8                 2400 ;
OFC8                 2401 ; End of SET DEFAULT parameters. Assume EOL.
OFC8                 2402 ;
OFC8                 2403
OFC8                 2404 170$:   $DS_CLI CLI$K_BR,      ENUF,     EOL           ; Enough input
OFCC                 2405
OFCC                 2406 ;
OFCC                 2407 ; End of SET DEFAULT
OFCC                 2408 ;
```

```
                     0FCC  2410          .SBTTL  DS$Cli Routine - Command Line Interpreter
                 00000000  2411          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000  2412 ;++
                     0000  2413 ; FUNCTIONAL DESCRIPTION:
                     0000  2414 ;
                     0000  2415 ;       This routine provides the basic command interpreter for the
                     0000  2416 ;       supervisor.
                     0000  2417 ;
                     0000  2418 ; CALLING SEQUENCE:
                     0000  2419 ;
                     0000  2420 ;       DS$CLI ()
                     0000  2421 ;
                     0000  2422 ; INPUT PARAMETERS:
                     0000  2423 ;
                     C000  2424 ;       AP      Address of R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP,PC,PSL
                     0000  2425 ;
                     0000  2426 ;
                     0000  2427 ; IMPLICIT INPUTS:
                     0000  2428 ;
                     0000  2429 ;       NONE
                     0000  2430 ;
                     0000  2431 ; OUTPUT PARAMETERS:
                     0000  2432 ;
                     0000  2433 ;       NONE
                     0000  2434 ;
                     0000  2435 ; IMPLICIT OUTPUTS:
                     0000  2436 ;
                     0000  2437 ;       NONE
                     0000  2438 ;
                     0000  2439 ; COMPLETION CODES:
                     0000  2440 ;
                     0000  2441 ;       NONE
                     0000  2442 ;
                     0000  2443 ; SIDE EFFECTS:
                     0000  2444 ;
                     0000  2445 ;       ALL COMMAND SUBROUTINES ARE ENTERED WITH R2 POINTING TO
                     0000  2446 ;       "DS$GL_CLIBASE".
                     0000  2447 ;--
```

```
                          003C  0000  2449          .ENTRY  DS$CLI,^M<R2,R3,R4,R5>
                                0002  2450 RCLI:
          00000000'EF    7C    0002  2451          clrq    L^g_adapter                          ; Clear adapter string          [48]
          00000000'EF    94    0008  2452          ClrB    L^ShowMemoryFlags                    ; Clear flags for show mem
       52 00000444'EF    DE    000E  2453          MOVAL   L^DS$GL_CLIBASE+CLI$K_SIZE,R2        ; START AT THE END
                   72    D4    0015  2461          CLRL    -(R2)                                ; AFTER THIS IT IS QUAD ALIGNED
       50 00000088 8F    D0    0017  2463          MOVL    #CLI$K_SIZE/8,R0                     ; NUMBER OF QUAD WORDS REMAINING
                                001E  2464
                   72    7C    001E  2465 10$:     CLRQ    -(R2)                                ; CLEAR QUAD
                FB 50    F5    0020  2466          SOBGTR  R0,10$                               ; CLEAR OTHER QUADWORDS
                                0023  2467
                                0023  2468 20$:     Br_If_Not_APT   CLI                          ; If not in APT mode, branch    [55]
    22 00000000'EF   00    E4    002B  2469          BBSC    #DS$V_CTRLC,DS$GL_FLAGS, 30$         ; Branch if ^C typed
          00000000'EF    16    0033  2470          JSB     APT_MSG                              ; WAIT FOR APT MESSAGE TO BE ACKED
          00000000'EF    16    0039  2471          JSB     KB_CHECK_APT                         ; CHECK APT
          0000FE04'EF    D5    003F  2472          TSTL    L^DSA$GL_APTCOM                      ; YES- IS THERE A COMMAND ?     [41]
                   DC    13    0045  2473          BEQL    20$                                  ; NO - WAIT UNTIL THERE IS ONE
    04 A2 00000000'EF    DE    0047  2474          MOVAL   L^APT_COM,CLI$L_COMMAND(R2)          ; SET APT COMMAND DISPATCH      [41]
          00000000'EF    16    004F  2475          JSB     APT_COM                              ; ENTER APT COMMAND PROCESSOR
                                0055  2476
                                0055  2477 30$:     Clear_CtrlO                                  ; Clear the ^O flag             [55]
                                005D  2478          $PRINT  #ds$k_type_ds_prompt, -              ; Print out the prompt          [48]
                                005D  2479                    #ds$k_printi, -                    ; .. use PRINTI                 [48]
                                005D  2480                    L^DS$GT_PROMPT                     ; Print 'DS> ' prompt, tells    [41]
                                0070  2481                                                       ;  apt we're done               [41]
                   90    11    0070  2482          BRB     RCLI                                 ; CHECK FOR ANOTHER COMMAND
```

```
                                0072  2484
                                0072  2485 CLI:
                                0072  2486          Br_If_CRD_AutoTest_Off  5$      ; If running under CRD Auto Test, branch[72]
                                007A  2487          Br_If_CRD_MenuTest_Off  5$      ; If running under CRD Menu Test, branch[72]
                                0082  2488          Br_If_CRD_MenuTest_On   5$      ; If running under CRD Menu Test, branch[72]
                          08    11    008A  2489          Brb     10$               ; If not CRD, branch around              [67]
                                008C  2490
                00000000'EF   16    008C  2491 5$:      Jsb     KB_Check          ; Check for any ^C's                     [67]
                          08    11    0092  2492          Brb     20$               ; Branch around clearing ^C flag         [58]
                                0094  2493
                                0094  2494 10$:     Clear_CtrlC                   ; Clear the ^C flag                      [58]
                                009C  2495
                00000000'EF   9F    009C  2496 20$:     PUSHAB  L^DS$GT_PROMPT    ; ADDRESS OF PROMPT                      [58]
                00000100 8F   DD    00A2  2497          PUSHL   #CLI$K_BUFSIZ     ; SIZE OF BUFFER
                      34 A2   DF    00A8  2498          PUSHAL  CLI$Q_BUFQWD(R2)  ; WHERE TO PUT LENGTH
                      3C A2   9F    00AB  2499          PUSHAB  CLI$T_BUFFER(R2)  ; WHERE TO PUT TEXT
          00000000'EF   04    FB    00AE  2500          CALLS   #4,L^DS_SUPERLINE ; GET SUPERVISOR LINE                    [41]
                                00B5  2501
                00000000'EF   D4    00B5  2502          CLRL    L^Line_count      ; Reinit # lines output for SET PAGE     [73]
                                00BB  2503
             50   0000'8F     B1    00BB  2504          CMPW    #SS$_ENDOFFILE,R0 ; EOF?
                          09    12    00C0  2505          BNEQ    70$               ; BRANCH IF NOT EOF
                                00C2  2506          $EXIT_S                   ; RETURN TO VMS COMMAND LEVEL
                                00CB  2507
                       03 50    E8    00CB  2508 70$:     BLBS    R0,700$           ; If no errors, continue                 [41]
                        FF31    31    00CE  2509          BRW     RCLI              ; If error on input, try again           [41]
                                00D1  2510
                                00D1  2511 700$:                                    ;                                        [41]
                          51    D5    00D1  2512          TSTL    R1                ; CHECK FOR NULL STRING.
                          03    12    00D3  2513          BNEQ    71$               ; Don't loop if not zero                 [40]
                        FF2A    31    00D5  2514          BRW     RCLI              ; Loop if zero                           [40]
                                00D8  2515
                                00D8  2516 71$:                                     ;                                        [40]
          38 A2         3C A2   DE    00D8  2517          MOVAL   CLI$T_BUFFER(R2), -
                                00DD  2518                    CLI$Q_BUFQWD+4(R2)
```

```
                                00DD  2520
                                00DD  2521 ;
                                00DD  2522 ; Parse the Supervisor line.
                                00DD  2523 ;
                                00DD  2524
                0000013D'EF  DF 00DD  2525          PUSHAL  L^CLI_ACTION            ; Call the special Supervisor          [45]
                00000180'EF  DF 00E3  2526          PUSHAL  L^COMMAND_TREE          ;  parser. The only difference between [45]
                          34 A2 7F 00E9  2527          PUSHAQ  CLI$Q_BUFQWD(R2)        ;  it and the regular DS$PARSE is that [45]
                00000000'9F  03 FB 00EC  2528          CALLS   #3, @#DSX$SUPERPARSE    ;  it does not return when EOL (i.e., [45]
                                00F3  2529                                          ;  it continues parsing until CLI$K_EXIT[45]
                                00F3  2530                                          ;  is seen).                           [45]
                                00F3  2531
                          3C 50 E8 00F3  2532          BLBS    R0, 90$                 ; If parsed successfully, branch       [45]
                                00F6  2533
                                00F6  2534 ;
                                00F6  2535 ; Parsed unsuccessfully. Check NOTNUF flag. If it is set,
                                00F6  2536 ; implies an illegal command (see ACT_INC_CMD). It it is
                                00F6  2537 ; clear, implies an incomplete command (see ACT_INC_CMD).
                                00F6  2538 ;
                                00F6  2539
          1A 00000000 E2  01 E0 00F6  2540          BBS     #CLI$V_NOTNUF, -        ; If NOTNUF set, branch                [45]
                                00FE  2541            L^CLI$L_FLAGS (R2),75$;                                       [45]
                                00FE  2542
                                00FE  2543 ;
                                00FE  2544 ; ILLEGAL COMMAND
                                00FE  2545 ;
                                00FE  2546
                00000444'EF  9F 00FE  2547          PUSHAB  L^Q_GOOD_CMD            ; Push desc. of good part              [43]
                0000001B'EF  9F 0104  2548          PUSHAB  L^T_ILLCMD              ; Push ILLEGAL COMMAND                 [43]
                          00 DD 010A  2549          pushl   #ds$k_printi            ; use PRINTI                           [48]
                          15 DD 010C  2550          pushl   #ds$k_type_command_err  ; Command error code                  [48]
                00000000'GF  04 FB 010E  2551          CALLS   #4, G^DSX$PRINT         ;                                      [48]
                        FEEA 31 0115  2552          BRW     RCLI
                                0118  2553
                                0118  2554 ;
                                0118  2555 ; INCOMPLETE COMMAND
                                0118  2556 ;
                                0118  2557
                00000444'EF  9F 0118  2558 75$:     PUSHAB  L^Q_GOOD_CMD            ; Push desc. of good part              [43]
                0000003A'EF  9F 011E  2559          PUSHAB  L^T_INCOMPLETE          ; Push INCOMPLETE COMMAND              [43]
                          00 DD 0124  2560          pushl   #ds$k_printi            ; use PRINTI                           [48]
                          15 DD 0126  2561          pushl   #ds$k_type_command_err  ; Command error code                  [48]
                00000000'GF  04 FB 0128  2562          CALLS   #4, G^DSX$PRINT         ;                                      [48]
                        FED0 31 012F  2563          BRW     RCLI
                                0132  2564
                                0132  2565 ;
                                0132  2566 ; No errors anywhere!
                                0132  2567 ;
                                0132  2568
                    50  04 A2 D0 0132  2569 90$:     MOVL    CLI$L_COMMAND(R2),R0    ; TO GET ADDRESS OF EXECUTION ROUTINE.
                          02 13 0136  2570          BEQL    99$                     ; SKIP CALL IF NULL
                          60 16 0138  2571          JSB     (R0)                    ; CALL FINAL ACTION ROUTINE
                        FEC5 31 013A  2572 99$:     BRW     RCLI                    ; CONTINUE COMMAND SCAN
```

```
                        .SBTTL  CLI Action Routines.
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THESE ROUTINES SAVE THE COMMAND LINE INFORMATION AS IT IS BEING PARSED.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       SUBROUTINE CALL FROM THE ''PARSE'' SERVICE ROUTINE.
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R0      = ACTION CODE FROM TREE.
                ;       R1      = DATA, IF CLI$K_NUM.
                ;       R8      = POINTER TO NEXT CHARACTER.
                ;
                ; IMPLICIT INPUTS:      NONE
                ;
                ; OUTPUT PARAMETERS:    NONE
                ;
                ; IMPLICIT OUTPUTS:     NONE
                ;
                ; COMPLETION CODES:     NONE
                ;
                ; SIDE EFFECTS:         NONE
                ;--
```

```
                        013D  2601
                        013D  2602 CLI_ACTION::
80'8F   00   50   8F     013D  2603          CASEB   R0,#0, #$$$
                00000000 0142  2604 10$:     $$N=0
                        0142  2605
                        0142  2606          CASEDEF NULL,\$$N                       ; Must be first     [43]
                        0144  2607          CASEDEF ILL_CMD,\$$N                    ;                   [43]
                        0146  2608          CASEDEF INC_CMD,\$$N                    ;                   [43]
                        0148  2609          CASEDEF CONTEXT,\$$N                    ;                   [43]
                        014A  2610          CASEDEF SUCCESS,\$$N                    ;                   [46]
                        014C  2611          CASEDEF FAILURE,\$$N                    ;                   [46]
                        014E  2612          CASEDEF ABORT,\$$N
                        0150  2613          CASEDEF ATTACH,\$$N
                        0152  2614          CASEDEF CLEAR,\$$N
                        C154  2615          CASEDEF CONTINUE,\$$N
                        0156  2616          CASEDEF DEATTACH,\$$N                   ;                   [48]
                        0158  2617          CASEDEF DEBUG,\$$N                      ;                   [59]
                        015A  2618          CASEDEF DEBUG_SWITCH,\$$N               ;                   [62]
                        015C  2619          CASEDEF DEFAULT_DBG,\$$N                ;                   [59]
                        015E  2620          CASEDEF DEPOSIT,\$$N
                        0160  2621          CASEDEF DIRECTORY,\$$N
                        0162  2622          CASEDEF EXAMINE,\$$N
                        0164  2623          CASEDEF EXIT,\$$N                       ;                   [46]
                        0166  2624          CASEDEF HELP,\$$N
                        0168  2625          CASEDEF LOAD,\$$N
                        016A  2626          CASEDEF NEXT_INST,\$$N
                        016C  2627          CASEDEF ENUF,\$$N
                        016E  2628          CASEDEF NOTNUF,\$$N
                        0170  2629          CASEDEF RUN,\$$N
                        0172  2630          CASEDEF SCRIPT,\$$N
                        0174  2631          CASEDEF SET,\$$N
                        0176  2632          CASEDEF SHOW,\$$N
                        0178  2633          CASEDEF Clear_CRD_Trace,\$$N            ;                   [65]
                        017A  2634          CASEDEF Set_CRD_Trace,\$$N              ;                   [65]
                        017C  2635          CASEDEF Set_CRD_Debug,\$$N              ;                   [69]
                        017E  2636          CASEDEF Clear_CRD_Debug,\$$N            ;                   [69]
                        0180  2637          CASEDEF Show_CRD_Trace,\$$N             ;                   [65]
                        0182  2638          CASEDEF Show_Calls,\$$N                 ;                   [70]
                        0184  2639          CASEDEF START,\$$N
                        0186  2640          CASEDEF SUMMARY,\$$N
                        0188  2641          CASEDEF CRD,\$$N                        ;                   [60]
                        018A  2642          CASEDEF FILESPEC,\$$N
                        018C  2643          CASEDEF FILEND,\$$N
                        018E  2644          CASEDEF OPTIONAL,\$$N                   ;                   [45]
                        0190  2645          CASEDEF REQUIRED,\$$N                   ;                   [45]
                        0192  2646          CASEDEF TEST,\$$N
                        0194  2647          CASEDEF LAST,\$$N
                        0196  2648          CASEDEF SUBTEST,\$$N
                        0198  2649          CASEDEF MMON,\$$N
                        019A  2650          CASEDEF MMOFF,\$$N
                        019C  2651          CASEDEF SHOWMM,\$$N
                        019E  2652          CASEDEF PREGN,\$$N
                        01A0  2653          CASEDEF SECTION,\$$N
                        01A2  2654          CASEDEF PASS,\$$N
                        01A4  2655          CASEDEF QA,\$$N                         ;                   [42]
                        01A6  2656          CaseDef SetEnforce,\$$N                 ;                   [57]
                        01A8  2657          CaseDef ClearEnforce,\$$N               ;                   [57]
```

```
01AA  2658          CASEDEF EVENT,\$$N
01AC  2659          CASEDEF FLAGS,\$$N
01AE  2660          CASEDEF BASE,\$$N
01B0  2661          CASEDEF ADDRESS,\$$N
01B2  2662          CASEDEF BREAK,\$$N
01B4  2663          CASEDEF ALL,\$$N
01B6  2664          CASEDEF DEFAULT,\$$N
01B8  2665          CASEDEF BELL,\$$N
01BA  2666          CASEDEF BINARY,\$$N                                     ;       [45]
01BC  2667          CASEDEF HALT,\$$N
01BE  2668          CASEDEF IE1,\$$N
01C0  2669          CASEDEF IE2,\$$N
01C2  2670          CASEDEF IE3,\$$N
01C4  2671          CASEDEF IES,\$$N
01C6  2672          CASEDEF LOOP,\$$N
01C8  2673          CASEDEF OPER,\$$N
01CA  2674          CASEDEF PROMPT,\$$N
01CC  2675          CASEDEF QUICK,\$$N
01CE  2676          CASEDEF SEARCH,\$$N
01D0  2677          CASEDEF TRACE,\$$N
01D2  2678          CASEDEF VERIFY,\$$N                                     ;       [45]
01D4  2679          CASEDEF QAEP,\$$N                                       ;       [42]
01D6  2680          CASEDEF QACL,\$$N                                       ;       [42]
01D8  2681          CASEDEF QATL,\$$N                                       ;       [42]
01DA  2682          CASEDEF QASL,\$$N                                       ;       [42]
01DC  2683          CASEDEF QAMP,\$$N                                       ;       [42]
01DE  2684          CASEDEF QADEF,\$$N                                      ;       [42]
01E0  2685          CASEDEF EFN,\$$N
01E2  2686          CASEDEF NEXT,\$$N
01E4  2687          CASEDEF PAGE,\$$N                                       ;       [71]
01E6  2688          CASEDEF WIDTH,\$$N                                      ;       [38]
01E8  2689          CASEDEF ASCII,\$$N
01EA  2690          CASEDEF REGN,\$$N
01EC  2691          CASEDEF REG12,\$$N
01EE  2692          CASEDEF REG13,\$$N
01F0  2693          CASEDEF REG14,\$$N
01F2  2694          CASEDEF REG15,\$$N
01F4  2695          CASEDEF REG16,\$$N
01F6  2696          CASEDEF REGP,\$$N
01F8  2697          CASEDEF BACKUP,\$$N
01FA  2698          CASEDEF ADVANCE,\$$N                                    ;       [45]
01FC  2699          CASEDEF DATA,\$$N
01FE  2700          CASEDEF LONG,\$$N
0200  2701          CASEDEF WORD,\$$N
0202  2702          CASEDEF BYTE,\$$N
0204  2703          CASEDEF HEX,\$$N
0206  2704          CASEDEF DEC,\$$N
0208  2705          CASEDEF OCT,\$$N
020A  2706          CASEDEF IF_RUN,\$$N
020C  2707          CASEDEF IF_REQUIRED,\$$N                                ;       [45]
020E  2708          CASEDEF IF_FILE_SPEC,\$$N                               ;       [45]
0210  2709          CASEDEF IF_ADAPTER,\$$N                                 ;       [48]
0212  2710          CASEDEF IF_REG_OR_ADR,\$$N                              ;       [43]
0214  2711          CASEDEF IFNOTUSER,\$$N                                  ;       [38]
0216  2712          CASEDEF IF_XDELTA,\$$N
0218  2713          CASEDEF XDELTA,\$$N
021A  2714          CASEDEF DEVICE,\$$N
```

```
                     021C  2715          CASEDEF BRIEF,\$$N                                              ;  [44]
                     021E  2716          CASEDEF BEGINADAPTER,\$$N                                       ;  [43]
                     0220  2717          CASEDEF ADAPTER,\$$N                                            ;  [43]
                     0222  2718          CASEDEF DESELECT,\$$N
                     0224  2719          CASEDEF SELECT,\$$N
                     0226  2720          CASEDEF SHOWSEL,\$$N
                     0228  2721          CASEDEF DO_CMD,\$$N
                     022A  2722          CASEDEF SETLOAD,\$$N
                     022C  2723          CASEDEF SHOWLOAD,\$$N
                     022E  2724          CASEDEF SHOWSUP,\$$N                                            ;  [44]
                     0230  2725          CASEDEF SHOWSTATUS,\$$N                                         ;  [39]
                     0232  2726          CASEDEF SHOWSECTIONS,\$$N                                       ;  [46]
                     0234  2727          CaseDef ShowMem,\$$N
                     0236  2728          CaseDef ShoMemMap,\$$N
                     C238  2729          CaseDef ShoMemBuf,\$$N
                     023A  2730          CaseDef ShoMemDat,\$$N
                     023C  2731          CaseDef ShoMemAll,\$$N
                     023E  2732          CASEDEF SETMEM,\$$N                                             ;  [75]
                     0240  2733          CASEDEF INITPCS,\$$N                                            ;  [77]
                     0242  2734          CASEDEF DIR_WIDE,\$$N                                           ;  [80]
                     0244  2735 ;        CASEDEF BEGINTIME,\$$N                                          ;  [76]
                     0244  2736 ;        CASEDEF TIME,\$$N                                               ;  [76]
                     0244  2737
           00000080  0244  2738          $$$=$$N-1
                     0244  2739
                     0244  2740 20$:     ERRSUP_S  MSGADR=L^T_PRGERR     ;*** INVALID ACTION CODE????   [41]
                  05 0259  2741          RSB
```

```
                                025A  2743          .SUBTITLE       Null and Context-Saving Action Routines
                                025A  2744 ;+
                                025A  2745 ; NULL ACTION ROUTINE.
                                025A  2746 ; This action routine must remain here. It is the action routine that is
                                025A  2747 ; done when the parser executes the 0 action routine.
                                025A  2748 ;-
                                025A  2749 ACT_NULL:
                             05 025A  2750          RSB                                 ; RETURN
                                025B  2751
                                025B  2752 ;+
                                025B  2753 ; ILL_CMD Action routine.
                                025B  2754 ;-
                                025B  2755 ACT_ILL_CMD:                                  ;                    [43]
                   06 62    01  E5  025B  2756          BBCC    #CLI$V_NOTNUF, -        ;                    [43]
```

```
                          025F  2758                      CLI$L_FLAGS (R2), ACT_CONTEXT ;                                [43]
                    04    025F  2759 10$:    BRB     ACT_CONTEXT                        ;                                [43]
                          0261  2760
                          0261  2761 ;+
                          0261  2762 ; INC_CMD Action routine.
                          0261  2763 ;-
                          0261  2764 ACT_INC_CMD:                                       ;                                [43]
                00 62     01    E3  0261  2765          BBCS    #CLI$V_NOTNUF, -        ;                                [43]
                          0265  2766                      CLI$L_FLAGS (R2), ACT_CONTEXT ;                                [43]
                          0265  2767
                          0265  2768 ;+
                          0265  2769 ; CONTEXT Action routine.
                          0265  2770 ; Branch to when error. Save descriptor of good portion of command line.
                          0265  2771 ;-
                          0265  2772
                          0265  2773 ACT_CONTEXT:                                      ;                                [43]
00000448'EF     38 A2     D0  0265  2774          MOVL    CLI$Q_BUFQWD+4(R2), -       ; Address of good portion        [43]
                          026D  2775                      Q_GOOD_CMD+4                ;                                [43]
   58   00000448'EF       C3  026D  2776          SUBL3   Q_GOOD_CMD+4, -             ;                                [43]
        00000444'EF           0274  2777                  R8, Q_GOOD_CMD              ; Length                         [43]
                    05    0279  2778          RSB                                   ; Return                         [43]
```

```
                                027A  2780          .SUBTITLE       Success, Failure, Notnuf, Enuf Action Routines
                                027A  2781 ;+
                                027A  2782 ; SUCCESS Action routine.
                                027A  2783 ;-
                                027A  2784 ACT_SUCCESS:
00 0000044C'EF    00    E3      027A  2785          BBCS    #0, B_SUCCESS, 10$      ; Indicate SUCCESS                  [46]
                          05    0282  2786 10$:     RSB
                                0283  2787
                                0283  2788 ;+
                                0283  2789 ; FAILURE Action routine.
                                0283  2790 ;-
                                0283  2791 ACT_FAILURE:
00 0000044C'EF    00    E5      0283  2792          BBCC    #0, B_SUCCESS, 10$      ; Indicate FAILURE                  [46]
                          05    028B  2793 10$:     RSB
                                028C  2794
                                028C  2795 ;+
                                028C  2796 ; NOTNUF ACTION ROUTINE.
                                028C  2797 ;-
                                028C  2798 ACT_NOTNUF:
                00 62     01    E3  028C  2799          BBCS    #CLI$V_NOTNUF, -        ; SET NOT ENOUGH FLAG.
                                0290  2800                    CLI$L_FLAGS(R2), 10$
                          05    0290  2801 10$:     RSB                             ; RETURN
                                0291  2802
                                0291  2803 ;+
                                0291  2804 ; ENUF Action routine.
                                0291  2805 ;-
                                0291  2806 ACT_ENUF:
                00 62     01    E4  0291  2807          BBSC    #CLI$V_NOTNUF, -        ; INDICATE SUFFICIENT INFO.
                                0295  2808                    CLI$L_FLAGS(R2), 10$
                          05    0295  2809 10$:     RSB                             ; RETURN.
```

```
                                0296  2811          .SUBTITLE       Commands A-C Action Routines
                                0296  2812 ;+
                                0296  2813 ; ATTACH COMMAND.
                                0296  2814 ;-
                                0296  2815 ACT_ATTACH:
            08 A2    59    D0   0296  2816          MOVL    R9,CLI$Q_FILE(R2)       ; SAVE LENGTH OF STRING
            0C A2    58    D0   029A  2817          MOVL    R8,CLI$Q_FILE+4(R2)     ; SAVE ADDRESS OF REMAINDER
    04 A2   00000000'EF    9E   029E  2818          MOVAB   L^DSV$ATTACH,CLI$L_COMMAND(R2) ; SET COMMAND PROCESSOR          [41]
                           05   02A6  2819          RSB
                                02A7  2820
                                02A7  2821 ;+
                                02A7  2822 ; ABORT COMMAND.
                                02A7  2823 ;-
                                02A7  2824 ACT_ABORT:
    04 A2   00000000'EF    DE   02A7  2825          MOVAL   L^VRABORT, -            ; LOAD COMMAND ROUTINE ADDRESS.        [41]
                                02AF  2826                    CLI$L_COMMAND(R2)
                           05   02AF  2827          RSB
                                02B0  2828
                                02B0  2829 ;+
                                02B0  2830 ; CLEAR COMMAND.
                                02B0  2831 ;-
                                02B0  2832 ACT_CLEAR:
    04 A2   00000000'EF    DE   02B0  2833          MOVAL   L^VRCLRFLG, -           ; LOAD COMMAND ROUTINE ADDRESS.        [41]
                                02B8  2834                    CLI$L_COMMAND(R2)
            00 62    02    E2   02B8  2835          BBSS    #CLI$V_CLEAR, -         ; SET CLR FLG.
                                02BC  2836                    CLI$L_FLAGS(R2), 10$
                           05   02BC  2837 10$:     RSB                             ; RETURN.
                                02BD  2838
                                02BD  2839 ;+
                                02BD  2840 ; CONTINUE COMMAND.
                                02BD  2841 ;-
                                02BD  2842 ACT_CONTINUE:
       00000000'EF    02    8A   02BD  2843          BICB2   #2,L^DEBUG$GB_FLAGS     ; CLEAR STEP FLAG                      [41]
    04 A2   000002CD'EF    9E   02C4  2844          MOVAB   10$,CLI$L_COMMAND(R2)   ; SET COMMAND ADDRESS
                           05   02CC  2845          RSB
                                02CD  2846 10$:
                                02CD  2847          $PRINT  #ds$k_type_command_out,-; Print command output                 [48]
                                02CD  2848                  #ds$k_printf, -         ; .. use PRINTF                        [48]
                                02CD  2849                  L^T_CONT, -             ; .. Continue message                  [48]
                                02CD  2850                  60(AP)                  ; .. Type PC                           [48]
            00000000'EF    16   02E3  2851          JSB     SCRIPT$CONT             ; Continue script if present
                                02E9  2852
                                02E9  2853 ACT_CONT::
                           04   02E9  2854 10$:     RET                             ; RETURN TO CLI'S CALLER
```

```
                                02EA  2856          .SUBTITLE       Commands D-E Action Routines
                                02EA  2857 ;+
                                02EA  2858 ; DEATTACH command
                                02EA  2859 ;-                                                                   [48]
                                02EA  2860 act_deattach:                        ;                               [48]
04 A2   00000000'EF     DE      02EA  2861          moval   L^dsv$deattach, -   ; Load command routine address [48]
                                02F2  2862                    cli$l_command(r2) ;                               [48]
                             05 02F2  2863          rsb                         ; Just return                   [48]
                                02F3  2864                                                                      [59]
                                02F3  2865 ;+                                                                   [59]
                                02F3  2866 ; DEBUG COMMAND                                                      [59]
                                02F3  2867 ;-                                                                   [59]
                                02F3  2868                                                                      [59]
                                02F3  2869 ACT_DEBUG:                           ;                               [59]
04 A2   00000000'EF     DE      02F3  2870          MOVAL   L^DSV$DEBUG,CLI$L_COMMAND(R2)   ; LOAD ADDR. OF DEBUG COMMAND [59]
                             05 02FB  2871          RSB                         ; RETURN                        [59]
                                02FC  2872
                                02FC  2873 ACT_DEFAULT_DBG::                    ; BUILD FILENAME DESC. FOR DEFAULT DBG [59]
08 A2   0000030D'EF     9A      02FC  2874          MOVZBL  DBG_NAME,CLI$Q_FILE(R2) ; GET CHARACTER COUNT           [59]
0C A2   0000030E'EF     DE      0304  2875          MOVAL   DBG_NAME+1, -       ; GET ADDRESS OF FILENAME STRING [59]
                                030C  2876                    CLI$Q_FILE+4(R2);                                 [59]
                             05 030C  2877          RSB                         ; RETURN                        [59]
47 55 42 45 44 53 44 56 00'     030D  2878 DBG_NAME:        .ASCIC /VDSDEBUG/   ; DEFAULT DEBUGGER NAME         [68]
                             08 030D
                                0316  2879
                                0316  2880 ;+
                                0316  2881 ; DEPOSIT COMMAND.
                                0316  2882 ;-
                                0316  2883 ACT_DEPOSIT:
04 A2   00000000'EF     DE      0316  2884          MOVAL   L^VRDEPOSIT, -      ; LOAD COMMAND ROUTINE ADDRESS. [41]
                                031E  2885                    CLI$L_COMMAND(R2)
     .      00 62   19  E2      031E  2886          BBSS    #CLI$V_DEPOSIT, -   ; SET DEPOSIT FLG.
                                0322  2887                    CLI$L_FLAGS(R2), 10$
                             05 0322  2888 10$:     RSB
                                0323  2889
                                0323  2890 ;+
                                0323  2891 ; DESELECT
                                0323  2892 ;-
                                0323  2893 ACT_DESELECT:
04 A2   00000000'EF     9E      0323  2894          MOVAB   L^DSV$DESELECT,CLI$L_COMMAND(R2) ; SET COMMAND ROUTINE [41]
                             05 032B  2895          RSB
                                032C  2896
                                032C  2897 ;
                                032C  2898 ; Directory command
                                032C  2899 ;-
                                032C  2900
                                032C  2901 ACT_DIRECTORY:
        00000000'EF     DE      032C  2902          MOVAL   L^DSV$DIRECTORY,-   ; Set dispatch address for command
              04 A2             0332  2903                    CLI$L_COMMAND(R2)
        00000000'EF     94      0334  2904          CLRB    L^WIDE_FLAG         ; WIDE_FLAG in DIRECTORY module [80]
                             05 033A  2905          RSB
                                033B  2906                                                                      [80]
                                033B  2907 ACT_DIR_WIDE:                        ;                               [80]
   00000000'EF   01     90      033B  2908          MOVB    #1,L^WIDE_FLAG      ; WIDE_FLAG in DIRECTORY module [80]
                             05 0342  2909          RSB                         ;                               [80]
                                0343  2910
                                0343  2911 ;+
```

```
                                0343  2912 ; EXAMINE COMMAND.
                                0343  2913 ;-
                                0343  2914 ACT_EXAMINE:
04 A2   00000000'EF    DE       0343  2915         MOVAL   L^VREXAMINE, -          ; LOAD COMMAND ROUTINE ADDRESS.          [41]
                                034B  2916                   CLI$L_COMMAND(R2)
 00 62    05           E2       034B  2917         BBSS    #CLI$V_EXAM, -          ; CHECK FOR EXAMINE.
                                034F  2918                   CLI$L_FLAGS(R2), 10$
                          05    034F  2919 10$:     RSB
                                0350  2920
                                0350  2921 ;+
                                0350  2922 ; EXIT Action routine.
                                0350  2923 ;-
                                0350  2924 ACT_EXIT:
04 A2   00000000'EF    DE       0350  2925         MOVAL   L^DSV$EXIT, -           ; Load EXIT routine address              [46]
                                0358  2926                   CLI$L_COMMAND (R2)    ;                                        [46]
                          05    0358  2927         RSB                             ; Return                                 [46]
```

```
                                0359  2929          .SUBTITLE       Commands H-R Action Routines
                                0359  2930 ;+
                                0359  2931 ; HELP Command action routine
                                0359  2932 ;-
                                0359  2933
                                0359  2934 ACT_HELP:
     00000000'EF    16          0359  2935          JSB     SCRIPT$FLUSH            ; Flush scripts if console command
     00000000'EF    16          035F  2936          JSB     DS_CLEANUP              ; Clean up program if necessary
50   00000000'EF    9E          0365  2937          MOVAB   HELPINFO,R0             ; Place to save pointers
      04 A0    58   D0          036C  2938          MOVL    R8,4(R0)                ; Save address
         60    59   D0          0370  2939          MOVL    R9,(R0)                 ; Save length
     00000000'EF    16          0373  2940          JSB     INIT_CONTEXT            ; Re-init stacks, PCB, ...
7E   00000000'EF    7D          0379  2941          MOVQ    HELPINFO,-(SP)          ; Restore descriptor
                6E  7F          0380  2942          PUSHAQ  (SP)                    ; and set pointer to it
00000000'EF    01   FB          0382  2943          CALLS   #1,L^DSX$HELP           ; Call the HELP routine
                8E  7C          0389  2944          CLRQ    (SP)+                   ; Remove descriptor
             06 50  E8          038B  2945          BLBS    R0,20$                  ; If no HELP error, continue
     00000000'EF    16          038E  2946          JSB     DSR$COMPLETION          ; Type error text
           FC69'    31          0394  2947 20$:     BRW     BEGIN                   ; Can't return--so start over
                                0397  2948
                                0397  2949 ;+
                                0397  2950 ; INITPCS                                                        begin   [77]
                                0397  2951 ;-
                                0397  2952
                                0397  2953 ACT_INITPCS:
   04 A2    9D'AF    9E          0397  2954          MOVAB   B^10$,CLI$L_COMMAND(R2) ; Load command routine address
                     05          039C  2955          RSB
                                 039D  2956
                                 039D  2957 10$:     BR_IF_NOT_USER  20$
      000000AB'EF    9F          03A5  2958          PUSHAB  L^T_EMESSG1             ;
                 01  DD          03AB  2959          pushl   #ds$k_printf            ; Print with PRINTF
                 15  DD          03AD  2960          pushl   #ds$k_type_command_err  ; Command error code
00000000'GF      03  FB          03AF  2961          CALLS   #3, G^DSX$PRINT         ; Print
                     05          03B6  2962          RSB
                                 03B7  2963
00000000'EF      00  FB          03B7  2964 20$:     CALLS   #0,DSV$INITPCS          ; Routine to do the work
                     05          03BE  2965          RSB
                                 03BF  2966
                                 03BF  2967 ;+
                                 03BF  2968 ; LOAD COMMAND.
                                 03BF  2969 ;-
                                 03BF  2970 ACT_LOAD:
04 A2    00000000'EF    DE       03BF  2971          MOVAL   L^DSV$LOAD, -           ; Load command routine address
                                 03C7  2972                    CLI$L_COMMAND(R2)
      62    40 8F    88          03C7  2973          BISB2   #1@CLI$V_LOAD, -
                                 03CB  2974                    CLI$L_FLAGS(R2)       ; SET LOAD COMMAND
   18 A2    0000'8F  3C          03CB  2975          MOVZWL  #DS$A_PRGBGN,CLI$L_ADDRESS(R2) ; DEFAULT LOAD ADDRESS
                     05          03D1  2976          RSB
                                 03D2  2977
                                 03D2  2978 ;+
                                 03D2  2979 ; NEXT Action routine.
                                 03D2  2980 ;-
                                 03D2  2981 ACT_NEXT_INST:
      1C A2    01    D0          03D2  2982          MOVL    #1, CLI$L_DATA(R2)      ; DEFAULT IS ONE STEP
00000000'EF    02    88          03D6  2983          BISB2   #2,L^DEBUG$GB_FLAGS     ; SET STEP FLAG                 [41]
   04 A2    FF08 CF  9E          03DD  2984          MOVAB   W^ACT_CONT,CLI$L_COMMAND(R2)  ; SET PROCESSING ADDRESS
                     05          03E3  2985          RSB
```

```
                                03E4  2986
                                03E4  2987 ;+
                                03E4  2988 ; RUN COMMAND.
                                03E4  2989 ;-
                                03E4  2990 ACT_RUN:
          00 62    15    E3     03E4  2991          BBCS    #CLI$V_RUN, -
                                03E8  2992            CLI$L_FLAGS (R2), 5$  ; Set RUN flag                                [45]
    04 A2    00000000'EF  DE     03E8  2993 5$:      MOVAL   L^DSV$RUN, -          ; LOAD COMMAND ROUTINE ADDRESS.       [41]
                                03F0  2994            CLI$L_COMMAND(R2)
          2C A2    01    D0     03F0  2995          MOVL    #1,CLI$L_PASS(R2)     ; Default pass count is 1
 03 0000FE00'EF    0E    E1     03F4  2996          BBC     #DSA$V_SEARCH,L^DSA$GL_FLAGS,10$ ; Branch if search not set [41]
             2C A2    D4     03FC  2997          CLRL    CLI$L_PASS(R2)        ; If in SEARCH, default to 0.
                      05     03FF  2998 10$:     RSB                           ; Return
```

```
                                0400  3000          .SUBTITLE       Commands S-Z Action Routines
                                0400  3001
                                0400  3002 ;+
                                0400  3003 ; SELECT
                                0400  3004 ;-
                                0400  3005 ACT_SELECT:
04 A2   00000000'EF    9E       0400  3006          MOVAB   L^DSV$SELECT,CLI$L_COMMAND(R2)  ; SET COMMAND ROUTINE          [41]
                       05       0408  3007          RSB
                                0409  3008
                                0409  3009 ;+
                                0409  3010 ; SET COMMAND.
                                0409  3011 ;-
                                0409  3012 ACT_SET:
04 A2   00000000'EF    DE       0409  3013          MOVAL   L^VRSETFLG, -           ; LOAD COMMAND ROUTINE ADDRESS.      [41]
                                0411  3014            CLI$L_COMMAND(R2)
        00 62    03    E2       0411  3015          BBSS    #CLI$V_SET, -           ; SET SET FLG.
                                0415  3016            CLI$L_FLAGS(R2), 10$
                       05       0415  3017 10$:     RSB                             ; RETURN.
                                0416  3018
                                0416  3019 ;+
                                0416  3020 ; SHOW COMMAND.
                                0416  3021 ;-
                                0416  3022 ACT_SHOW:
        00 62    04    E2       0416  3023          BBSS    #CLI$V_SHOW, -          ; SET SHOW FLG.
                                041A  3024            CLI$L_FLAGS(R2), 10$
                       05       041A  3025 10$:     RSB                             ; RETURN.
                                041B  3026
                                041B  3027 ;+
                                041B  3028 ; START COMMAND.
                                041B  3029 ;-
                                041B  3030 ACT_START:
        00 62    15    E5       041B  3031          BBCC    #CLI$V_RUN, -           ;                                    [45]
                                041F  3032            CLI$L_FLAGS (R2), 5$  ; Clear RUN flag                             [45]
04 A2   00000000'EF    DE       041F  3033 5$:      MOVAL   L^VRSTART, -            ; LOAD COMMAND ROUTINE ADDRESS.      [41]
                                0427  3034            CLI$L_COMMAND(R2)
        2C A2    01    D0       0427  3035          MOVL    #1,CLI$L_PASS(R2)       ; Default pass count is 1
03 0000FE00'EF   0E    E1       042B  3036          BBC     #DSA$V_SEARCH,L^DSA$GL_FLAGS,10$ ; Branch if search not set  [41]
           2C A2       D4       0433  3037          CLRL    CLI$L_PASS(R2)          ; If in SEARCH, default to 0.
                       05       0436  3038 10$:     RSB                             ; Return
                                0437  3039
                                0437  3040 ;+
                                0437  3041 ; SCRIPT COMMAND.
                                0437  3042 ;-
                                0437  3043 ACT_SCRIPT:
04 A2   00000000'EF    DE       0437  3044          MOVAL   L^SCRIPT$OPEN, -        ; LOAD COMMAND ROUTINE ADDRESS.      [41]
                                043F  3045            CLI$L_COMMAND(R2)
                       05       043F  3046          RSB
                                0440  3047
                                0440  3048 ;+
                                0440  3049 ; SUMMARY COMMAND.
                                0440  3050 ;-
                                0440  3051 ACT_SUMMARY:
04 A2   00000000'EF    DE       0440  3052          MOVAL   L^VRSUMMARY, -          ; LOAD COMMAND ROUTINE ADDRESS.      [41]
                                0448  3053            CLI$L_COMMAND(R2)
                       05       0448  3054          RSB
```

```
                                0449  3056          .SubTitle       CRD command
                                0449  3057
                                0449  3058 ;+                                                                    [60]
                                0449  3059 ; CRD COMMAND.                                                        [60]
                                0449  3060 ;-                                                                    [60]
                                0449  3061                                                                       [60]
                                0449  3062 ACT_CRD:                                    ;                         [72]
                                0449  3063          Br_If_CRD_MenuTest_Off  100$     ; If already running under CRD, branch [72]
                                0451  3064          Br_If_CRD_MenuTest_On   100$     ; If already running under CRD, branch [72]
                                0459  3065          Br_If_CRD_AutoTest_Off  100$     ; If already running under CRD, branch [72]
                          0065  31     0461  3066          BRW     150$              ; Determine CRD mode              [72]
                                0464  3067
                                0464  3068 10$:
                                0464  3069
                00000000'EF     16    C464  3070          Jsb     L^DS_Cleanup      ; Clean up the old diagnostic     [67]
                                046A  3071
                                046A  3072          Br_If_User      15$             ; If User_Mode, branch around     [67]
             00000000'EF  00    FB    0472  3073          Calls   #0, L^MapFree     ; Map free memory                 [67]
                                0479  3074
                00000000'EF     D4    0479  3075 15$:     ClrL    L^DS$GA_DS_Ctrl_C_First ; Clear the DS's 1st ^C handler [67]
                00000000'EF     D4    047F  3076          ClrL    L^DS$GA_DS_Ctrl_C_Second; Clear the DS's 2nd ^C handler [66]
                00000000'EF     D4    0485  3077          ClrL    L^DS$L_UserCntrlC       ; Cleat the user's ^C handler [66]
                                048B  3078
             00000000'EF  00    E4    048B  3079          Bbsc    #DS$V_CtrlC, -          ; Clear the ^C pending bit [66]
                             00       0492  3080                    L^DS$GL_Flags, 20$  ;                          [67]
                                0493  3081
             00000000'EF  18    E2    0493  3082 20$:     Bbss    #DS$V_DisablCC, -       ; Disable ^C's for now (set the bit) [67]
                             00       049A  3083                    L^DS$GL_Flags, 30$  ;                          [67]
                                049B  3084
             00000000'EF  00    FB    049B  3085 30$:     Calls   #0, L^DSR$Load_CRD      ; Load the CRD image file  [67]
                                04A2  3086
                                04A2  3087          ;+
                                04A2  3088          ; Call the CRD's initialization routine. If it returns success, we [67]
                                04A2  3089          ; are on our way. If it returns failure, print a message and go call [67]
                                04A2  3090          ; the DSR$Unload_CRD routine so that the CRD region can be deallocated. [67]
                                04A2  3091          ;-
                                04A2  3092
                          00    DD    04A2  3093          PushL   #0                      ; Push 0 as param-4        [63]
                          00    DD    04A4  3094          PushL   #0                      ; Push 0 as param-3        [63]
                          00    DD    04A6  3095          PushL   #CRD$K_CRD_Initialization ; Push Initialization as param-2 [63]
                          00    DD    04A8  3096          PushL   #CRD$K_Hook_Point       ; Push Hook_Point as param-1 [63]
             00000000'FF  04    FB    04AA  3097          CallS   #4, -                   ; Call the CRD$DS_Interface routine [63]
                                04B1  3098                    @L^DS$GA_CRD_DS_Interface ; ... to initialize CRD    [63]
                       66 50    E8    04B1  3099          BlbS    R0, 300$                ; If success, branch and exit [67]
                                04B4  3100
                                04B4  3101          ;+
                                04B4  3102          ; Come to here if: 1. Already running CRD and the CRD command is [67]
                                04B4  3103          ; issued, or, 2. If the CRD imitialization routine return failure. [67]
                                04B4  3104          ; This should never happen, but . . .                        [67]
                                04B4  3105          ;-
                                04B4  3106
                                04B4  3107 100$:    $PRINT  #DS$K_TYPE_COMMAND_ERR,-; Print illegal command      [63]
                                04B4  3108                    #DS$K_PRINTI, -       ; .. use Printi              [63]
                                04B4  3109                    L^GT_Menu_Error       ; .. the format string       [63]
                                04C7  3110
                          48    11    04C7  3111          Brb     200$              ; Branch to the error part     [67]
                                04C9  3112
```

```
                        04C9  3113          ;+
                        04C9  3114          ; Come to here if CRD is not active.                                    [72]
                        04C9  3115          ; Check to see if special debugging is required.                        [72]
                        04C9  3116          ;-
                        04C9  3117
                        04C9  3118 150$:    CASE    L^CRD$GL_CRD_Command_Debug,LIMIT=#1,TYPE=B,DISPLIST=<-
                        04C9  3119                              160$,-
                        04C9  3120                              165$ -
                        04C9  3121                              >
                        04D5  3122          BR_If_User      190$            ;No special debugging; error if
                        04DD  3123                                          ;User Mode                          [72]
                        04DD  3124          Set_CRD_MenuTest_Off            ;Perform CRD MENU Offline Pkg.      [72]
                FF7C  31 04E5  3125          BRW 10$                        ;Continue                           [72]
                        04E8  3126 160$:    Set_CRD_MenuTest_Off            ;If this =1: Run CRD MENU Offline Package
                FF71  31 04F0  3127          BRW     10$
                        04F3  3128 165$:    Set_CRD_Autotest_Off            ;If this =2: Run CRD AUTO Offline Package
                FF66  31 04FB  3129          BRW 10$
                        04FE  3130
                        04FE  3131          ;+
                        04FE  3132          ; Print an error message since command not permitted in User mode.      [72]
                        04FE  3133          ;
                        04FE  3134          ;-
                        04FE  3135
                        04FE  3136 190$:    $Print  #DS$K_Type_Command_Err,-        ; Print illegal command . . .   [72]
                        04FE  3137                    #DS$K_PrintI, -           ; . . . use PrintI              [64]
                        04FE  3138                    L^T_User_Mode_Menu_Error  ; . . . the format string       [64]
                        0511  3139
                  00    DD  0511  3140 200$:    PushL   #0                      ; Don't bother returning to here[67]
                        0513  3141                                              ; . . . if the image is loaded  [67]
      00000000'EF   01    FB  0513  3142          Calls   #1, L^DSR$Unload_CRD    ; Unload the CRD image if it    [67]
                        051A  3143                                              ; . . . was loaded.             [67]
                        051A  3144
                    05  051A  3145 300$:    Rsb                             ; Return to caller              [67]
```

```
                                051B  3147          .SUBTITLE       Filespec, Optional, Required Action Routines
                                051B  3148 ;+
                                051B  3149 ; FILESPEC ACTION ROUTINE.
                                051B  3150 ;-
                                051B  3151 ACT_FILESPEC:
                0C A2    68     DE  051B  3152          MOVAL   (R8), CLI$Q_FILE+4(R2)  ; SAVE CHARACTER POINTER.
                   08 A2       D4  051F  3153          CLRL    CLI$Q_FILE(R2)          ; CLEAR LENGTH OF STRING
                               05  0522  3154          RSB                             ; RETURN
                                0523  3155
                                0523  3156 ;+
                                0523  3157 ; FILEND ACTION ROUTINE.
                                0523  3158 ;-
                                0523  3159 ACT_FILEND:
                58   0C A2      C3  0523  3160          SUBL3   CLI$Q_FILE+4(R2), -     ; CALC CHARACTER COUNT.
                      08 A2          0527  3161                    R8, CLI$Q_FILE(R2)
                               05  0529  3162          RSB                             ; RETURN.
                                052A  3163
                                052A  3164 ;+
                                052A  3165 ; OPTIONAL Action routine.
                                052A  3166 ;-
                                052A  3167 ACT_OPTIONAL:                            ;                               [45]
                00 62    00     E5  052A  3168          BBCC    #CLI$V_REQUIRED, -      ; Clear required bit            [45]
                                052E  3169                    CLI$L_FLAGS (R2), 10$ ;                               [45]
                               05  052E  3170 10$:     RSB                             ; Return                        [45]
                                052F  3171
                                052F  3172 ;+
                                052F  3173 ; REQUIRED Action routine.
                                052F  3174 ;-
                                052F  3175 ACT_REQUIRED:                            ;                               [45]
                00 62    00     E3  052F  3176          BBCS    #CLI$V_REQUIRED, -      ; Set required bit              [45]
                                0533  3177                    CLI$L_FLAGS (R2), 10$ ;                               [45]
                               05  0533  3178 10$:     RSB                             ; Return                        [45]
```

```
                                0534  3180          .SUBTITLE       Start and Run Qualifier Action Routines
                                0534  3181
                                0534  3182 ;+                                                                          [62]
                                0534  3183 ; /DEBUG ACTION ROUTINE                                                    [62]
                                0534  3184 ;-                                                                          [62]
                                0534  3185 ACT_DEBUG_SWITCH:                        ;                                [62]
                40000000 8F   C8  0534  3186          BISL2   #1@DSA$V_LOAD_DEBUGGER,-;SET INDICATOR SO DS WILL LOAD    [62]
                0000FE00'EF       053A  3187                  DSA$GL_FLAGS        ;  DEBUGGER LATER                [62]
                              05  053F  3188          RSB                         ;RETURN                          [62]
                                0540  3189
                                0540  3190 ;+
                                0540  3191 ; /PASSES ACTION ROUTINE.
                                0540  3192 ;-
                                0540  3193 ACT_PASS:
                 2C A2    5A  D0  0540  3194          MOVL    R10, CLI$L_PASS(R2)     ; SAVE NUMBER OF PASSES.
                              05  0544  3195          RSB                             ; RETURN.
                                0545  3196
                                0545  3197 ;+
                                0545  3198 ; /QA Action routine
                                0545  3199 ;-
                                0545  3200 ACT_QA:
                 00 62    07  E2  0545  3201          BBSS    #CLI$V_QA, -                    ; Set QA flag on        [42]
                                0549  3202                      CLI$L_FLAGS(R2),10$         ;                       [42]
                              05  0549  3203 10$:     RSB                             ;                       [42]
                                054A  3204
                                054A  3205 ;+
                                054A  3206 ; /SECTION ACTION ROUTINE.
                                054A  3207 ;-
                                054A  3208 ACT_SECTION:
              10 A2    58    51  C3  054A  3209          SUBL3   R1, R8, -               ; CALCULATE SECTION NAME SIZE.
                                054F  3210                    CLI$Q_SECTION(R2)
                  14 A2    51  D0  054F  3211          MOVL    R1, CLI$Q_SECTION+4(R2) ; ADDRESS OF SECTION NAME.
                              05  0553  3212          RSB                             ; RETURN.
                                0554  3213
                                0554  3214 ;+
                                0554  3215 ; /SUBTEST ACTION ROUTINE.
                                0554  3216 ;-
                                0554  3217 ACT_SUBTEST:
                  28 A2    5A  D0  0554  3218          MOVL    R10, CLI$L_SUBT(R2)     ; SAVE SUBTEST NUMBER.
                              05  0558  3219          RSB                             ; RETURN.
                                0559  3220
                                0559  3221 ;+
                                0559  3222 ; /TEST ACTION ROUTINE.
                                0559  3223 ;-
                                0559  3224 ACT_TEST:
                  20 A2    5A  D0  0559  3225          MOVL    R10, CLI$L_TEST(R2)     ; SAVE FIRST TEST NUMBER.
                              05  055D  3226          RSB                             ; RETURN.
                                055E  3227
                                055E  3228 ;+
                                055E  3229 ; LAST ACTION ROUTINE.
                                055E  3230 ;-
                                055E  3231 ACT_LAST:
                  24 A2    5A  D0  055E  3232          MOVL    R10, CLI$L_LAST(R2)     ; SAVE LAST TEST NUMBER.
                              05  0562  3233          RSB                             ; RETURN.
                                0563  3234
                                0563  3235 ;+
                                0563  3236 ; BEGINTIME ACTION ROUTINE for /TIME switch     ;                      [76]
```

```
0563  3237 ;-
0563  3238 ;
0563  3239 ;ACT_BEGINTIME:                                                    ;                                                  [76]
0563  3240 ;       MOVAL   (R8),CLI$Q_TIME+4(R2)                             ; Save current address of time string [76]
0563  3241 ;       CLRL    CLI$Q_TIME(R2)                                    ; Clear the length field                   [76]
0563  3242 ;       RSB                                                       ;                                          [76]
0563  3243 ;
0563  3244 ;+
0563  3245 ; /TIME ACTION ROUTINE                                            ;                                          [76]
0563  3246 ;-
0563  3247 ;
0563  3248 ;ACT_TIME:                                                       ;                                          [76]
0563  3249 ;;      PUSHR   #^M<R1,R2,R3,R4,R5,R6>                            ; MOVC3 destroys R0-R5, LOCC               [76]
0563  3250 ;                                                                 ; destroys R0,R1, don't need R0            [76]
0563  3251 ;                                                                 ; Need R6 to hold R2 temporarily           [76]
0563  3252 ;       MOVL    R2,R6                                             ; Save R2 (DS$GL_CLIBASE)                  [76]
0563  3253 ;       SUBL3   CLI$Q_TIME+4(R6),R8, -                            ; Calculate length of time...              [76]
0563  3254 ;               CLI$Q_TIME(R6)                                    ; string                                   [76]
0563  3255 ;
0563  3256 ;       LOCC    #^A"-",CLI$Q_TIME(R6), -                          ; a "-" means a value for day              [76]
0563  3257 ;               @CLI$Q_TIME+4(R6)                                 ; was typed.                               [76]
0563  3258 ;       BNEQ    20$                                               ; Branch if found                          [76]
0563  3259 ;
0563  3260 ;                                                                 ; Not found, copy string to                [76]
0563  3261 ;                                                                 ; stack, in order to insert "0 "           [76]
0563  3262 ;10$:   TSTW    -(SP)                                             ; Allocate space for "0 "                  [76]
0563  3263 ;       SUBL2   CLI$Q_TIME(R6),SP                                 ; Allocate space for string                [76]
0563  3264 ;       MOVW    #^A"0 ",(SP)                                      ; "0 " means 0 days                        [76]
0563  3265 ;       MOVC3   CLI$Q_TIME(R6), -                                 ; Move the string from the CLI             [76]
0563  3266 ;               @CLI$Q_TIME+4(R6),2(SP)                           ; data structure to the stack              [76]
0563  3267 ;       MOVAL   (SP),@CLI$Q_TIME+4(R6)                            ; Set new address of string               [76]
0563  3268 ;       ADDL2   #2,CLI$Q_TIME(R6)                                 ; Add 2 to the string length              [76]
0563  3269 ;       $BINTIM_S CLI$Q_TIME(R6),L^BIN_TIME ; Convert ASCII to binary time[76]
0563  3270 ;       ADDL2   CLI$Q_TIME(R6),SP                                 ; Deallocate stack space                   [76]
0563  3271 ;       POPR    #^M<R1,R2,R3,R4,R5,R6>                            ; These registers matter                   [76]
0563  3272 ;       BRB     40$
0563  3273 ;
0563  3274 ;20$:   CMPL    CLI$Q_TIME+4(R6),R1                               ; "-" first char in string?                [76]
0563  3275 ;       BNEQ    30$                                               ; Branch if not                            [76]
0563  3276 ;       DECL    CLI$Q_TIME(R6)                                    ; Decr string count (rid of "-")           [76]
0563  3277 ;       INCL    CLI$Q_TIME+4(R6)                                  ; Incr string address                      [76]
0563  3278 ;       BRB     10$                                               ; Go pad in a "0 "                         [76]
0563  3279 ;
0563  3280 ;                                                                 ; Here if a day value preceded             [76]
0563  3281 ;                                                                 ; the dash                                 [76]
0563  3282 ;30$:   MOVB    #^A" ",(R1)                                       ; Change the "-" to " "                    [76]
0563  3283 ;       $BINTIM_S CLI$Q_TIME(R6),L^BIN_TIME ; Convert ascii to binary time[76]
0563  3284 ;       POPR    #^M<R1,R2,R3,R4,R5,R6>                            ; These registers matter                   [76]
0563  3285 ;
0563  3286 ;40$:   BLBS    R0,50$                                            ; Branch if success                        [76]
0563  3287 ;       MOVL    #1,R0                                             ; Failure, set R0 for CLI$K_BIF            [76]
0563  3288 ;       RSB                                                       ;                                          [76]
0563  3289 ;
0563  3290 ;50$:   CLRL    R0                                                ; Success, set R0 for CLI$K_BIF            [76]
0563  3291 ;       RSB                                                       ;                                          [76]
```

```
                                0563  3293          .SUBTITLE       Event and Flags Action Routines
                                0563  3294
                                0563  3295 ;+
                                0563  3296 ; Set/Clear Enforce action routines.  Note that 'SET ENFORCE'
                                0563  3297 ; actually CLEARS the flag, and 'CLEAR ENFORCE' SETS the flag.
                                0563  3298 ; The physical flag is actually 'inhibit enforce'.
                                0563  3299 ;-
                                0563  3300 .Enable LSB                                                          [57]
                                0563  3301 Act_SetEnforce:                        ;                           [57]
                          50    94  0563  3302          ClrB    R0                ; Clear value               [57]
                          03    11  0565  3303          BrB     Common_Enforce    ;                           [57]
                                0567  3304 Act_ClearEnforce:                      ;                           [57]
                        50  01    90  0567  3305          MovB    #1, R0            ; Set value                 [57]
                                056A  3306 Common_Enforce:                        ;                           [57]
          00000000'EF   50    90  056A  3307          MovB    R0, L^Ds$GB_Inhibit_Naming    ; transfer value    [57]
                    04  A2    D4  0571  3308          ClrL    Cli$L_Command(R2) ; Make sure no command set  [57]
                                05  0574  3309          Rsb                       ;                           [57]
                                0575  3310 .Disable LSB
                                0575  3311
                                0575  3312 ;+
                                0575  3313 ; EVENT ACTION ROUTINE.
                                0575  3314 ;-
                                0575  3315 ACT_EVENT:
                    00 62   08  E2  0575  3316          BBSS    #CLI$V_EVENT, -           ; SET 'EVENT FLAG' FLAG.
                                0579  3317                    CLI$L_FLAGS(R2), 10$
                    0A 62   03  E1  0579  3318 10$:     BBC     #CLI$V_SET, -             ; BR IF SET COMMAND.
                                057D  3319                    CLI$L_FLAGS(R2), 20$
          04 A2 00000000'EF     DE  057D  3320          MOVAL   L^VRSETEF, -              ; COMMAND.          [41]
                                0585  3321                    CLI$L_COMMAND(R2)
                          16    11  0585  3322          BRB     40$
                    0A 62   02  E1  0587  3323 20$:     BBC     #CLI$V_CLEAR, -           ; BR IF CLEAR COMMAND.
                                058B  3324                    CLI$L_FLAGS(R2), 30$
          04 A2 00000000'EF     DE  058B  3325          MOVAL   L^VRCLREF, -              ; COMMAND.          [41]
                                0593  3326                    CLI$L_COMMAND(R2)
                          08    11  0593  3327          BRB     40$
          04 A2 00000000'EF     DE  0595  3328 30$:     MOVAL   L^VRSHOWEF, -             ; COMMAND.          [41]
                                059D  3329                    CLI$L_COMMAND(R2)
                                05  059D  3330 40$:     RSB                       ; RETURN.
                                059E  3331
                                059E  3332 ;+
                                059E  3333 ; FLAGS ACTION ROUTINE.
                                059E  3334 ;-
                                059E  3335 ACT_FLAGS:
                    10 62   08  E0  059E  3336          BBS     #CLI$V_EVENT, -           ; 'EVENT FLAG' FLAG SET ?
                                05A2  3337                    CLI$L_FLAGS(R2), 20$      ; YES, THEN DO NOTHING HERE.
                    00 62   09  E2  05A2  3338          BBSS    #CLI$V_FLAGS, -           ; SET 'FLAGS' FLAG.
                                05A6  3339                    CLI$L_FLAGS(R2), 10$
                    08 62   04  E1  05A6  3340 10$:     BBC     #CLI$V_SHOW, -            ; BR IF NOT SHOW COMMAND.
                                05AA  3341                    CLI$L_FLAGS(R2), 20$
          04 A2 00000000'EF     DE  05AA  3342          MOVAL   L^VRSHOWFLG, -            ; COMMAND.          [41]
                                05B2  3343                    CLI$L_COMMAND(R2)
                                05  05B2  3344 20$:     RSB                       ; RETURN.
```

```
                                05B3  3346          .SUBTITLE       Base, Address, Break Action Routines
                                05B3  3347
                                05B3  3348 ;+
                                05B3  3349 ; BASE ACTION ROUTINE.
                                05B3  3350 ;-
                                05B3  3351 ACT_BASE:
                     09 62  04  E1  05B3  3352          BBC     #CLI$V_SHOW, -          ; Branch if not SHOW command          [43]
                                05B7  3353                    CLI$L_FLAGS (R2), 10$ ;                                       [43]
                04 A2  00000000'EF  DE  05B7  3354          MOVAL   L^VRSHOWBASE, -         ; "SHOW BASE" command                 [43]
                                05BF  3355                    CLI$L_COMMAND (R2)    ;                                       [43]
                                05  05BF  3356          RSB                             ;                                     [43]
                04 A2  00000000'EF  DE  05C0  3357 10$:     MOVAL   L^VRSETBASE, -          ; "SET BASE" COMMAND.                 [41]
                                05C8  3358                    CLI$L_COMMAND(R2)
                                05  05C8  3359          RSB                             ; RETURN.
                                C5C9  3360
                                05C9  3361 ;+
                                05C9  3362 ; ADDRESS ACTION ROUTINE.
                                05C9  3363 ;-
                                05C9  3364 ACT_ADDRESS:
                     18 A2  5A  D0  05C9  3365          MOVL    R10, CLI$L_ADDRESS(R2)
             00 00000000 E2  0B  E2  05CD  3366          BBSS    #CLI$V_ADR, -           ; Got address
                                05D5  3367                    L^CLI$L_FLAGS (R2), 10$
                                05  05D5  3368 10$:     RSB
                                05D6  3369
                                05D6  3370 ;+
                                05D6  3371 ; BREAK ACTION ROUTINE.
                                05D6  3372 ;-
                                05D6  3373 ACT_BREAK:
                     00 62  0A  E2  05D6  3374          BBSS    #CLI$V_BREAK, -         ; SET 'BREAKPOINT' FLAG.
                                05DA  3375                    CLI$L_FLAGS(R2), 10$
                     09 62  03  E1  05DA  3376 10$:     BBC     #CLI$V_SET, -           ; BR IF NOT 'SET' COMMAND.
                                05DE  3377                    CLI$L_FLAGS(R2), 20$
                04 A2  00000000'EF  DE  05DE  3378          MOVAL   L^VRSETBRK, -           ; 'SET BREAK' COMMAND.                [41]
                                05E6  3379                    CLI$L_COMMAND(R2)
                                05  05E6  3380          RSB                             ; RETURN.
                     09 62  02  E1  05E7  3381 20$:     BBC     #CLI$V_CLEAR, -         ; BR IF NOT 'CLEAR' COMMAND.
                                05EB  3382                    CLI$L_FLAGS(R2), 30$
                04 A2  00000000'EF  DE  05EB  3383          MOVAL   L^VRCLRBRK, -           ; 'CLEAR BREAK' COMMAND.              [41]
                                05F3  3384                    CLI$L_COMMAND(R2)
                                05  05F3  3385          RSB                             ; RETURN.
                04 A2  00000000'EF  DE  05F4  3386 30$:     MOVAL   L^VRSHOWBRK, -          ; 'SHOW BREAK' COMMAND.               [41]
                                05FC  3387                    CLI$L_COMMAND(R2)
                                05  05FC  3388          RSB
```

```
                                                05FD  3390          .SUBTITLE       All, Default Action Routines
                                                05FD  3391
                                                05FD  3392 ;+
                                                05FD  3393 ; ALL ACTION ROUTINE.
                                                05FD  3394 ;-
                                                05FD  3395 ACT_ALL:
                    10 A2    00      D2         05FD  3396          MCOML   #0, CLI$L_DATA(R2)
                       04 A2 01      D5         0601  3397          TSTL    CLI$L_COMMAND(R2)
                             01      13         0604  3398          BEQL    10$
                             05      05         0606  3399          RSB
                    0A 62    03      E1         0607  3400 10$:     BBC     #CLI$V_SET, -           ; BR IF NOT 'SET' COMMAND.
                                                0608  3401                    CLI$L_FLAGS(R2), 20$
04 A2   00000000'EF          DE                 0608  3402          MOVAL   L^VRSETFLG, -           ;                                       [41]
                                                0613  3403                    CLI$L_COMMAND(R2)
                             08      11         0613  3404          BRB     30$
04 A2   00000000'EF          DE                 0615  3405 20$:     MOVAL   L^VRCLRFLG, -           ;                                       [41]
                                                061D  3406                    CLI$L_COMMAND(R2)
                             05      05         061D  3407 30$:     RSB                             ; RETURN.
                                                061E  3408
                                                061E  3409 ;+
                                                061E  3410 ; DEFAULT ACTION ROUTINE.
                                                061E  3411 ;-
                                                061E  3412 ACT_DEFAULT:
                    16 62    09      E0         061E  3413          BBS     #CLI$V_FLAGS, -         ; If "SET FLAGS", branch.               [43]
                                                0622  3414                    CLI$L_FLAGS(R2), 20$
                                                0622  3415
                    09 62    04      E1         0622  3416          BBC     #CLI$V_SHOW, -          ; Branch if not SHOW command            [43]
                                                0626  3417                    CLI$L_FLAGS (R2), 10$ ;                                 [43]
04 A2   00000000'EF          DE                 0626  3418          MOVAL   L^VRSHOWDFLT, -         ; "SHOW DEFAULT" command                [43]
                                                062E  3419                    CLI$L_COMMAND (R2)    ;                                 [43]
                             05      05         062E  3420          RSB                           ;                                 [43]
                                                062F  3421
04 A2   00000000'EF          DE                 062F  3422 10$:     MOVAL   L^VRSETDFLT, -          ; "SET DEFAULT" command.                [41]
                                                0637  3423                    CLI$L_COMMAND(R2)
                             05      05         0637  3424          RSB                             ;                                       [43]
                                                0638  3425
                    00 62    0C      E2         0638  3426 20$:     BBSS    #CLI$V_DEFAULT, -       ; Set DEFAULT flag for                  [43]
                                                063C  3427                    CLI$L_FLAGS(R2), 30$  ;   "SET FLAGS DEFAULT" command.  [43]
                             05      05         063C  3428 30$:     RSB                           ;                                 [43]
```

```
                                063D  3430          .Subtitle       Show Calls Routine                                      ;       [70]
                                063D  3431
                                063D  3432 ;+                                                                                      [70]
                                063D  3433 ; Show Calls action routine.                                                            [70]
                                063D  3434 ;-                                                                                      [70]
                                063D  3435
                                063D  3436 Act_Show_Calls:                                                                 ;       [70]
        04 A2   00000000'EF  DE 063D  3437          MovAL   L^VRShow_Calls, -       ; SHOW CALLS command routine               [70]
                                0645  3438                    CLI$L_Command (R2)    ;                                          [70]
          1C A2   00         D0 0645  3439          MovL    #0, CLI$L_Data(R2)      ; 0 => show all call frames                [70]
                             05 0649  3440          Rsb                             ; Return to caller                         [70]
```

```
                                 064A  3442          .SubTitle       CRDTrace Action Routines                        ;       [65]
                                 064A  3443
                                 064A  3444 ;+
                                 064A  3445 ; Set/Clear/Show CRDTrace action routines.                                 [65]
                                 064A  3446 ;-
                                 064A  3447
                                 064A  3448 Act_Set_CRD_Trace:                          ;                              [65]
0000FE00'EF   11   E3           064A  3449          Bbcs    #DSA$V_CRD_Trace, -        ; Set the bit on               [65]
                00              0651  3450                    L^DSA$GL_Flags, 10$    ;                              [65]
                                0652  3451
                35   11         0652  3452 10$:   BrB     Common_CRD_Trace_Debug     ; Branch to the common stuff   [69]
                                0654  3453
                                0654  3454 Act_Clear_CRD_Trace:                        ;                              [65]
0000FE00'EF   11   E5           0654  3455          Bbcc    #DSA$V_CRD_Trace, -        ; Clear the bit - set it off   [65]
                00              0658  3456                    L^DSA$GL_Flags, 10$    ;                              [65]
                                065C  3457
                28   11         065C  3458 10$:   BrB     Common_CRD_Trace_Debug     ; Branch to the common stuff   [69]
                                065E  3459
                                065E  3460 Act_Show_CRD_Trace:                         ;                              [65]
50   000000EC'EF   9E           065E  3461          MovAB   L^T_On, R0                 ; Assume the bit is turned on  [65]
0000FE00'EF   11   E0           0665  3462          Bbs     #DSA$V_CRD_Trace, -        ; If the bit is turned on, . . .[65]
                07              066C  3463                    L^DSA$GL_Flags, -      ; . . . then branch to 10$, . . .[65]
                                066D  3464                    10$                    ; . . . and print it.          [65]
50   000000EF'EF   9E           066D  3465          MovAB   L^T_Off, R0                ; The bit is off.              [65]
                                0674  3466
                                0674  3467 10$:   $Print  #DS$K_Type_Command_Out,-; Print a message giving the status of [65]
                                0674  3468                    #DS$K_PrintI,       -; . . . the CRDTrace flag.       [65]
                                0674  3469                    L^T_CRD_Trace_Output,-;                               [65]
                                0674  3470                    R0                 ; . . . "on" or "off".         [65]
                                0689  3471
                                0689  3472 Common_CRD_Trace_Debug:                    ; The common stuff             [69]
00000004   E2   D4              0689  3473          ClrL    L^Cli$L_Command(R2)        ; Make sure no command routine is set  [65]
                05              068F  3474          Rsb                                ; Return to the DS$Cli routine [65]
                                0690  3475
                                0690  3476 Act_Set_CRD_Debug:                        ; Set the CRD debug bit - for CRD debug [69]
                                0690  3477                                           ; ... statements              [69]
00000000'EF   01   D0           0690  3478          MovL    #1, L^DS$GL_CRD_Debug      ; 1 => debug is on.            [69]
                B1   11         0697  3479          Brb     Act_Set_CRD_Trace          ; Turn Trace on also           [69]
                                0699  3480
                                0699  3481 Act_Clear_CRD_Debug:                      ; Clear the CRD debug bit - for CRD [69]
                                0699  3482                                           ; ... debug statements        [69]
00000000'EF   D4                0699  3483          ClrL    L^DS$GL_CRD_Debug          ; 0 => no debug statements     [69]
                B3   11         069F  3484          Brb     Act_Clear_CRD_Trace        ; Clear Trace also             [69]
```

```
                        06A1  3486          .SubTitle       Bell, Binary, Halt, IEx Action Routines
                        06A1  3487
                        06A1  3488 ;+
                        06A1  3489 ; BELL ACTION ROUTINE.
                        06A1  3490 ;-
                        06A1  3491 ACT_BELL:
```

```
                00 1C A2    03    E2  06A1  3493          BBSS    #DSA$V_BELL, -
                                      06A6  3494                    CLI$[_DATA(R2), 10$
```

```
                    05  06A6  3496 10$:    RSB
                        06A7  3497
                        06A7  3498 ;+
```

```
                        06A7  3500 ; BINARY ACTION ROUTINE.
                        06A7  3501 ;-
                        06A7  3502 ACT_BINARY:                                                  ;       [45]
00 1C A2    01    E2    06A7  3503          BBSS    #DSA$V_BINARY, -                           ;       [45]
                        06AC  3504            CLI$L_DATA(R2), 10$                              ;       [45]
                  05    06AC  3505 10$:     RSB                                                ;       [45]
                        06AD  3506
                        06AD  3507 ;+
                        06AD  3508 ; HALT ACTION ROUTINE.
                        06AD  3509 ;-
                        06AD  3510 ACT_HALT:
00 1C A2    00    E2    06AD  3511          BBSS    #DSA$V_HALT, -
                        06B2  3512            CLI$L_DATA(R2), 10$
                  05    06B2  3513 10$:     RSB
                        C6B3  3514
                        06B3  3515 ;+
                        06B3  3516 ; IE1 ACTION ROUTINE.
                        06B3  3517 ;-
                        06B3  3518 ACT_IE1:
00 1C A2    04    E2    06B3  3519          BBSS    #DSA$V_IE1, -
                        06B8  3520            CLI$L_DATA(R2), 10$
                  05    06B8  3521 10$:     RSB
                        06B9  3522
                        06B9  3523 ;+
                        06B9  3524 ; IE2 ACTION ROUTINE.
                        06B9  3525 ;-
                        06B9  3526 ACT_IE2:
00 1C A2    05    E2    06B9  3527          BBSS    #DSA$V_IE2, -
                        06BE  3528            CLI$L_DATA(R2), 10$
                  05    06BE  3529 10$:     RSB
                        06BF  3530
                        06BF  3531 ;+
                        06BF  3532 ; IE3 ACTION ROUTINE.
                        06BF  3533 ;-
                        06BF  3534 ACT_IE3:
00 1C A2    06    E2    06BF  3535          BBSS    #DSA$V_IE3, -
                        06C4  3536            CLI$L_DATA(R2), 10$
                  05    06C4  3537 10$:     RSB
                        06C5  3538
                        06C5  3539 ;+
                        06C5  3540 ; IES ACTION ROUTINE.
                        06C5  3541 ;-
                        06C5  3542 ACT_IES:
00 1C A2    07    E2    06C5  3543          BBSS    #DSA$V_IES, -
                        06CA  3544            CLI$L_DATA(R2), 10$
                  05    06CA  3545 10$:     RSB
```

```
                                06CB  3547          .SUBTITLE       Loop, Oper, Prompt Quick Action Routines
                                06CB  3548
                                06CB  3549 ;+
                                06CB  3550 ; LOOP ACTION ROUTINE.
                                06CB  3551 ;-
                                06CB  3552 ACT_LOOP:
00 1C A2    02    E2            06CB  3553          BBSS    #DSA$V_LOOP, -
                                06D0  3554                    CLI$L_DATA(R2), 10$
                          05    06D0  3555 10$:     RSB
                                06D1  3556
                                06D1  3557 ;+
                                06D1  3558 ; OPER ACTION ROUTINE.
                                06D1  3559 ;-
                                06D1  3560 ACT_OPER:
00 1C A2    0C    E2            06D1  3561          BBSS    #DSA$V_OPER, -
                                06D6  3562                    CLI$L_DATA(R2), 10$
                          05    06D6  3563 10$:     RSB
                                06D7  3564
                                06D7  3565 ;+
                                06D7  3566 ; PROMPT ACTION ROUTINE.
                                06D7  3567 ;-
                                06D7  3568 ACT_PROMPT:
00 1C A2    0D    E2            06D7  3569          BBSS    #DSA$V_PROMPT, -
                                06DC  3570                    CLI$L_DATA(R2), 10$
                          05    06DC  3571 10$:     RSB
                                06DD  3572
                                06DD  3573 ;+
                                06DD  3574 ; QUICK ACTION ROUTINE.
                                06DD  3575 ;-
                                06DD  3576 ACT_QUICK:
00 1C A2    08    E2            06DD  3577          BBSS    #DSA$V_QUICK, -
                                06E2  3578                    CLI$L_DATA(R2), 10$
                          05    06E2  3579 10$:     RSB
```

```
                                06E3  3581          .SUBTITLE       Search, Trace, Verify Action Routines
                                06E3  3582
                                06E3  3583 ;+
                                06E3  3584 ; SEARCH action routine
                                06E3  3585 ;-
                                06E3  3586 ACT_SEARCH:
00 1C A2    0E    E2            06E3  3587          BBSS    #DSA$V_SEARCH, -
                                06E8  3588                    CLI$L_DATA(R2),10$
                          05    06E8  3589 10$:     RSB
                                06E9  3590
                                06E9  3591 ;+
                                06E9  3592 ; TRACE ACTION ROUTINE.
                                06E9  3593 ;-
                                06E9  3594 ACT_TRACE:
00 1C A2    0A    E2            06E9  3595          BBSS    #DSA$V_TRACE, -
                                06EE  3596                    CLI$L_DATA(R2), 10$
                          05    06EE  3597 10$:     RSB
                                06EF  3598
                                06EF  3599 ;+
                                06EF  3600 ; VERIFY ACTION ROUTINE.
                                06EF  3601 ;-
                                06EF  3602 ACT_VERIFY:                                                  ;       [45]
00 1C A2    09    E2            06EF  3603          BBSS    #DSA$V_VERIFY, -                            ;       [45]
                                06F4  3604                    CLI$L_DATA(R2), 10$                       ;       [45]
                          05    06F4  3605 10$:     RSB                                                 ;       [45]
```

```
                                06F5  3607          .SUBTITLE       Set/Show Page, Width Action Routine
                                06F5  3608
                                06F5  3609 ;+
                                06F5  3610 ; SET/SHOW PAGE action routine
                                06F5  3611 ;-
                                06F5  3612 ACT_PAGE:                                                    ;                                       [71]
                 0D 62    04    E0  06F5  3613          BBS     #CLI$V_SHOW, -                         ; If SHOW command                       [71]
                                06F9  3614                   CLI$L_FLAGS(R2), 10$                   ;  branch                               [71]
                 12 62    03    E1  06F9  3615          BBC     #CLI$V_SET, -                          ; If not SET command                    [71]
                                06FD  3616                   CLI$L_FLAGS(R2), 20$                   ;  no command                           [71]
          04 A2  00000000'EF    9E  06FD  3617          MOVAB   DSV$SETPAGE, -                         ;                                       [71]
                                0705  3618                   CLI$L_COMMAND(R2)                      ; Set command address                   [71]
                             05 0705  3619          RSB                                            ;                                       [71]
          04 A2  00000000'EF    9E  0706  3620 10$:     MOVAB   DSV$SHOWPAGE, -                        ; Set SHOW command address              [71]
                                070E  3621                   CLI$L_COMMAND(R2)                      ;                                       [71]
                             05 070E  3622          RSB                                            ;                                       [71]
                       04 A2    D4  070F  3623 20$:     CLRL    CLI$L_COMMAND(R2)                      ; Make sure no routine address          [71]
                             05 0712  3624          RSB                                            ;                                       [71]
                                0713  3625
                                0713  3626 ;+
                                0713  3627 ; SET/SHOW WIDTH action routine
                                0713  3628 ;-
                                0713  3629 ACT_WIDTH:                                                  ;                                       [38]
                 0D 62    04    E0  0713  3630          BBS     #CLI$V_SHOW, -                         ; If SHOW command                       [38]
                                0717  3631                   CLI$L_FLAGS(R2), 10$                   ;  branch                               [38]
                 12 62    03    E1  0717  3632          BBC     #CLI$V_SET, -                          ; If not SET command                    [38]
                                071B  3633                   CLI$L_FLAGS(R2), 20$                   ;  no command                           [38]
          04 A2  00000000'EF    9E  071B  3634          MOVAB   VRSETWIDTH, -                          ;                                       [38]
                                0723  3635                   CLI$L_COMMAND(R2)                      ; Set command address                   [38]
                             05 0723  3636          RSB
          04 A2  00000000'EF    9E  0724  3637 10$:     MOVAB   VRSHOWWIDTH, -                         ; Set SHOW command address              [38]
                                072C  3638                   CLI$L_COMMAND(R2)                      ;                                       [38]
                             05 072C  3639          RSB
                       04 A2    D4  072D  3640 20$:     CLRL    CLI$L_COMMAND(R2)                      ; Make sure no routine address          [38]
                             05 0730  3641          RSB                                            ;                                       [38]
```

```
                        0731  3643          .SUBTITLE       Set/Show QA Action Routines
                        0731  3644
                        0731  3645 ;                                                                                [42]
                        0731  3646 ; SET/SHOW QAxxxxxxxxx flags Action routine                                      [42]
                        0731  3647 ;   This is patterned after the SET/SHOW width action routine above            [42]
                        0731  3648 ;                                                                                [42]
                        0731  3649 ACT_QAEP:                                                    ;                  [42]
18 62    1B    E2       0731  3650          BBSS    #CLI$V_QAERRORPRINTS, -                     ; Set the bit on   [42]
                        0735  3651                          CLI$L_FLAGS(R2), QA_COMMON          ;   regardless    [42]
         16    11       0735  3652          BRB     QA_COMMON                                   ; Branch to common stuff[42]
                        0737  3653                                                                                  [42]
                        0737  3654 ACT_QACL:                                                    ;                  [42]
12 62    1C    E2       0737  3655          BBSS    #CLI$V_QACKLOOPLOOPS, -                     ; Set the bit on   [42]
                        073B  3656                          CLI$L_FLAGS(R2), QA_COMMON          ;   regardless    [42]
         10    11       073B  3657          BRB     QA_COMMON                                   ; Branch to common stuff[42]
                        073D  3658                                                                                  [42]
                        073D  3659 ACT_QATL:                                                    ;                  [42]
0C 62    1D    E2       073D  3660          BBSS    #CLI$V_QATESTLOOPS, -                       ; Set the bit on   [42]
                        0741  3661                          CLI$L_FLAGS(R2), QA_COMMON          ;   regardless    [42]
         0A    11       0741  3662          BRB     QA_COMMON                                   ; Branch to common stuff[42]
                        0743  3663                                                                                  [42]
                        0743  3664 ACT_QASL:                                                    ;                  [42]
06 62    1E    E2       0743  3665          BBSS    #CLI$V_QASUBTESTLOOPS, -                    ; Set the bit on   [42]
                        0747  3666                          CLI$L_FLAGS(R2), QA_COMMON          ;   regardless    [42]
         04    11       0747  3667          BRB     QA_COMMON                                   ; Branch to common stuff[42]
                        0749  3668                                                                                  [42]
                        0749  3669 ACT_QAMP:                                                    ;                  [42]
00 62    1F    E2       0749  3670          BBSS    #CLI$V_QAMULTIPLEPASS, -                    ; Set the bit on   [42]
                        074D  3671                          CLI$L_FLAGS(R2), QA_COMMON          ;   regardless    [42]
                        074D  3672                                                              ; Cont onto common stuff[42]
                        074D  3673                                                                                  [42]
                        074D  3674 QA_COMMON:                                                   ;                  [42]
         0D 62    04    E0       074D  3675          BBS     #CLI$V_SHOW, -                      ; If SHOW, branch to 10$[42]
                        0751  3676                          CLI$L_FLAGS(R2), 10$                ;   If not, ...   [42]
         12 62    03    E1       0751  3677          BBC     #CLI$V_SET, -                       ; If not SET or SHOW,  [42]
                        0755  3678                          CLI$L_FLAGS(R2), 20$                ;   branch to 20$  [42]
                        0755  3679                                                              ; If SET, ...      [42]
04 A2    00000000'EF    9E       0755  3680          MOVAB   VRSETQA, -                          ; Move address of SETQA [42]
                        075D  3681                          CLI$L_COMMAND(R2)                   ;   routine to command [42]
                        075D  3682                                                              ;   address      [42]
                        05       075D  3683          RSB
04 A2    00000000'EF    9E       075E  3684 10$:     MOVAB   VRSHOWQA, -                         ; Move address of SHOWQA[42]
                        0766  3685                          CLI$L_COMMAND(R2)                   ;   routine to command [42]
                        0766  3686                                                              ;   address
                        05       0766  3687          RSB
              04 A2     D4       0767  3688 20$:     CLRL    CLI$L_COMMAND(R2)                   ; No routine address   [42]
                        05       076A  3689          RSB                                         ;                  [42]
```

```
                               076B             .SUBTITLE      QADef Action Routine
                               076B
                               076B  ACT_QADEF:                                                    ;                       [42]
             DE 62    0C   E2  076B             BBSS     #CLI$V_DEFAULT, -                         ; Set the DEFAULT bit    [42]
                               076F                            CLI$L_FLAGS(R2), QA_COMMON          ;   on regardless        [42]
                               076F                                                                ; The DEFAULT bit isn't [42]
                               076F                                                                ;   really for QA, but  [42]
                               076F                                                                ;   will use it anyway! [42]
                               076F
             0D 62    04   E0  076F             BBS      #CLI$V_SHOW, -                            ; If SHOW, branch to 10$[42]
                               0773                            CLI$L_FLAGS(R2), 10$                ;   If not, ...          [42]
             12 62    03   E1  0773             BBC      #CLI$V_SET, -                             ; If not SET or SHOW,    [42]
                               0777                            CLI$L_FLAGS(R2), 20$                ;   branch to 20$        [42]
                               0777                                                                ; If SET, ...            [42]
   04 A2    00000000'EF    9E  0777             MOVAB    VRSETQA, -                                ; Move address of SETQA  [42]
                               077F                            CLI$L_COMMAND(R2)                   ;   routine to command   [42]
                               077F                                                                ;   address              [42]
                           05  077F             RSB
                               0780
   04 A2    00000000'EF    9E  0780  10$:       MOVAB    VRSHOWQA, -                               ; Move address of SHOWQA[42]
                               0788                            CLI$L_COMMAND(R2)                   ;   routine to command   [42]
                               0788                                                                ;   address
                           05  0788             RSB
                    04 A2  D4  0789  20$:       CLRL     CLI$L_COMMAND(R2)                         ; No routine address     [42]
                           05  078C             RSB                                                ;                        [42]
```

```
                                078D  3717          .SUBTITLE       Efn, Next, Ascii Action Routines
                                078D  3718
                                078D  3719 ;+
                                078D  3720 ; EFN ACTION ROUTINE.
                                078D  3721 ;-
                                078D  3722 ACT_EFN:
                          5A    D5  078D  3723 10$:     TSTL    R10                     ; CHECK FOR EFN 0.
                          0A    13  078F  3724          BEQL    30$
                       17 5A    D1  0791  3725          CMPL    R10, #23                ; CHECK FOR IN RANGE.
                          05    1A  0794  3726          BGTRU   30$
                    11 1C A2    5A  E3  0796  3727 20$:     BBCS    R10, CLI$L_DATA(R2), 40$
                                079B  3728 30$:                                         ; Print illegal event flag number       [45]
                                079B  3729          PUSHAB  L^T_ILLEFN              ;                                       [45]
                 00000061'EF    9F  079B  3730          pushl   #ds$k_printi            ; Use printi                            [48]
                          00    DD  07A1  3731          pushl   #ds$k_type_command_err  ; Command error                         [48]
                          15    DD  07A3  3732          CALLS   #3, G^DSX$PRINT         ;                                       [48]
            00000000'GF   03    FB  07A5  3733
                                07AC  3734          ;BBCS   #CLI$V_ERROR, -         ; Illegal event flag number            [45]
                                07AC  3735          ;         CLI$L_FLAGS(R2),40$   ;                                       [45]
                                07AC  3736
                                05  07AC  3737 40$:     RSB                             ; RETURN
                                07AD  3738
                                07AD  3739 ;+
                                07AD  3740 ; NEXT ACTION ROUTINE.
                                07AD  3741 ;-
                                07AD  3742 ACT_NEXT:
                       30 A2    5A  D0  07AD  3743          MOVL    R10, CLI$L_NEXT(R2)     ; SAVE NEXT COUNT.
                                05  07B1  3744          RSB                             ; RETURN
                                07B2  3745
                                07B2  3746 ;+
                                07B2  3747 ; ASCII ACTION ROUTINE.
                                07B2  3748 ;-
                                07B2  3749 ACT_ASCII:
                    00 62       13  E2  07B2  3750          BBSS    #CLI$V_ASCII, -
                                07B6  3751                    CLI$L_FLAGS(R2), 10$
                                05  07B6  3752 10$:     RSB                             ; RETURN.
```

```
                                07B7  3754          .SUBTITLE       Register Action Routines
                                07B7  3755
                                07B7  3756 ;+
                                07B7  3757 ; REGISTER ACTION ROUTINES.
                                07B7  3758 ;-
                                07B7  3759 ACT_REGN:
                00 62   14   E2 07B7  3760          BBSS    #CLI$V_REG, -
                                07B8  3761            CLI$L_FLAGS(R2), 10$
                        05      07B8  3762 10$:     RSB                             ; RETURN.
                                07BC  3763
                                07BC  3764 ACT_REG12:
                18 A2   0C   D0 07BC  3765          MOVL    #12, CLI$L_ADDRESS(R2)  ; AP.
                        0A   11 07C0  3766          BRB     ACT_REGM
                                07C2  3767 ACT_REG13:
                18 A2   0D   D0 07C2  3768          MOVL    #13, CLI$L_ADDRESS(R2)  ; FP.
                        04   11 07C6  3769          BRB     ACT_REGM
                                07C8  3770 ACT_REG14:
                18 A2   0E   D0 07C8  3771          MOVL    #14, CLI$L_ADDRESS(R2)  ; SP.
                                07CC  3772 ACT_REGM:
                        59   D5 07CC  3773          TSTL     R9                     ; CHECK FOR LAST CHARACTER.
                        1B   13 07CE  3774          BEQL    ACT_BACKUP              ; BACK UP IF NO MORE.
                        05      07D0  3775          RSB                             ; RETURN.
                                07D1  3776
                                07D1  3777 ACT_REG15:
                18 A2   0F   D0 07D1  3778          MOVL    #15, CLI$L_ADDRESS(R2)  ; PC.
                        04   11 07D5  3779          BRB     ACT_REGP
                                07D7  3780
                                07D7  3781 ACT_REG16:
                18 A2   10   D0 07D7  3782          MOVL    #16, CLI$L_ADDRESS(R2)  ; PSL.
                                07DB  3783 ACT_REGP:
                00 62   14   E2 07DB  3784          BBSS    #CLI$V_REG, -
                                07DF  3785            CLI$L_FLAGS(R2), ACT_REGNUF
                                07DF  3786 ACT_REGNUF:
                00 62   08   E2 07DF  3787          BBSS    #CLI$V_ADR, -
                                07E3  3788            CLI$L_FLAGS(R2), 10$
                        05      07E3  3789 10$:     RSB                             ; RETURN.
                                07E4  3790
                                07E4  3791 ;+
                                07E4  3792 ; PROCESSOR REGISTER ACTION ROUTINES
                                07E4  3793 ;-
                                07E4  3794 ACT_PREGN:
                00 62   1A   E2 07E4  3795          BBSS    #CLI$V_PREG,CLI$L_FLAGS(R2),10$ ; Set command type bit
                    FDDE    31  07E8  3796 10$:     BRW     ACT_ADDRESS                     ; Store register code
```

```
                        07EB  3798          .SUBTITLE       Backup, Advance, Data Action Routines
                        07EB  3799
                        07EB  3800 ;+
                        07EB  3801 ; BACKUP ACTION ROUTINE.
                        07EB  3802 ;-
                        07EB  3803 ACT_BACKUP:
                58   D7 07EB  3804          DECL    R8                      ; Backup one character in the
                59   D6 07ED  3805          INCL    R9                      ; line. Increment the remaining
                        07EF  3806                                          ; character count.
                     05 07EF  3807          RSB                             ; Return
                        07F0  3808
                        07F0  3809 ;+
                        07F0  3810 ; ADVANCE Action routine.
                        07F0  3811 ;-
                        C7F0  3812 ACT_ADVANCE:
                58   D6 07F0  3813          INCL    R8                      ; Advance one character in      [45]
                59   D7 07F2  3814          DECL    R9                      ; line. Decrement the remaining [45]
                        07F4  3815                                          ; character count.              [45]
                     05 07F4  3816          RSB                             ; Return                        [45]
                        07F5  3817
                        07F5  3818 ;+
                        07F5  3819 ; DATA ACTION ROUTINE.
                        07F5  3820 ;-
                        07F5  3821 ACT_DATA:
             1C A2   5A D0 07F5  3822          MOVL    R10, CLI$L_DATA(R2)     ; SAVE THE DATA.
                     05 07F9  3823          RSB                             ; RETURN.
```

```
                                07FA  3825          .SUBTITLE       Long, Word, Byte, Hex, Dec, Oct Action Routines
                                07FA  3826
                                07FA  3827 ;+
                                07FA  3828 ; LONG ACTION ROUTINE.
                                07FA  3829 ;-
                                07FA  3830 ACT_LONG:
                00 62   0F  E3  07FA  3831          BBCS    #CLI$V_LONG, -
                                07FE  3832                    CLI$L_FLAGS(R2), 10$
                            05  07FE  3833 10$:     RSB                             ; RETURN.
                                07FF  3834
                                07FF  3835 ;+
                                07FF  3836 ; WORD ACTION ROUTINE.
                                07FF  3837 ;-
                                07FF  3838 ACT_WORD:
                00 62   0E  E3  C7FF  3839          BBCS    #CLI$V_WORD, -
                                0803  3840                    CLI$L_FLAGS(R2), 10$
                            05  0803  3841 10$:     RSB                             ; RETURN.
                                0804  3842
                                0804  3843 ;+
                                0804  3844 ; BYTE ACTION ROUTINE.
                                0804  3845 ;-
                                0804  3846 ACT_BYTE:
                00 62   0D  E3  0804  3847          BBCS    #CLI$V_BYTE, -
                                0808  3848                    CLI$L_FLAGS(R2), 10$
                            05  0808  3849 10$:     RSB                             ; RETURN.
                                0809  3850
                                0809  3851 ;+
                                0809  3852 ; HEX ACTION ROUTINE.
                                0809  3853 ;-
                                0809  3854 ACT_HEX:
                00 62   12  E3  0809  3855          BBCS    #CLI$V_HEX, -
                                080D  3856                    CLI$L_FLAGS(R2), 10$
                            05  080D  3857 10$:     RSB                             ; RETURN.
                                080E  3858
                                080E  3859 ;+
                                080E  3860 ; DEC ACTION ROUTINE.
                                080E  3861 ;-
                                080E  3862 ACT_DEC:
                00 62   10  E3  080E  3863          BBCS    #CLI$V_DEC, -
                                0812  3864                    CLI$L_FLAGS(R2), 10$
                            05  0812  3865 10$:     RSB                             ; RETURN.
                                0813  3866
                                0813  3867 ;+
                                0813  3868 ; OCT ACTION ROUTINE.
                                0813  3869 ;-
                                0813  3870 ACT_OCT:
                00 62   11  E3  0813  3871          BBCS    #CLI$V_OCT, -
                                0817  3872                    CLI$L_FLAGS(R2), 10$
                            05  0817  3873 10$:     RSB                             ; RETURN.
```

```
                                0818  3875          .SUBTITLE       IF Action Routines
                                0818  3876
                                0818  3877 ;+
                                0818  3878 ; IF_RUN ACTION ROUTINE.
                                0818  3879 ;
                                0818  3880 ;       Returns:
                                0818  3881 ;               1 => RUN command AND no file-spec has been gotten yet
                                0818  3882 ;               0 => Either:
                                0818  3883 ;                       Not the RUN command
                                0818  3884 ;                    OR the file-spec has already been gotten
                                0818  3885 ;-
                                0818  3886 ACT_IF_RUN:
                          50    D4  0818  3887          CLRL    R0                              ; Assume negative.
                07 62     15    E1  081A  3888          BBC     #CLI$V_RUN, -
                                    C81E  3889                    CLI$L_FLAGS(R2), 10$      ; If not RUN, return with 0.
                      08 A2     D5  081E  3890          TSTL    CLI$Q_FILE (R2)             ; Check the char. count for the file-spec.
                         02     12  0821  3891          BNEQ    10$                         ; If > 0, => file-spec is there.
                         50     D6  0823  3892          INCL    R0                          ; RUN and no file-spec found.
                                05  0825  3893 10$:     RSB                                 ; Return.
                                    0826  3894
                                    0826  3895 ;+
                                    0826  3896 ; IF_REQUIRED Action routine.
                                    0826  3897 ;
                                    0826  3898 ;       Returns:
                                    0826  3899 ;               1 => Something was required for the command
                                    0826  3900 ;               0 => Something was optional for the command
                                    0826  3901 ;-
                                    0826  3902 ACT_IF_REQUIRED:                             ;                               [45]
                          50    D4  0826  3903          CLRL    R0                          ;                               [45]
                02 62     00    E1  0828  3904          BBC     #CLI$V_REQUIRED, -          ;                               [45]
                                    082C  3905                    CLI$L_FLAGS (R2), 10$   ;                               [45]
                          50    D6  082C  3906          INCL    R0                          ;                               [45]
                                05  082E  3907 10$:     RSB                                 ;                               [45]
                                    082F  3908
                                    082F  3909 ;+
                                    082F  3910 ; IF_FILE_SPEC Action routine.
                                    082F  3911 ;
                                    082F  3912 ;       Returns:
                                    082F  3913 ;               1 => A file-spec is present
                                    082F  3914 ;               0 => No file-spec present
                                    082F  3915 ;-
                                    082F  3916 ACT_IF_FILE_SPEC:                            ;                               [45]
                          50    D4  082F  3917          CLRL    R0                          ; Assume no file-spec present   [45]
                      08 A2     D5  0831  3918          TSTL    CLI$Q_FILE (R2)             ; See if the char-count is > zero [45]
                         02     13  0834  3919          BEQL    10$                         ; If = 0, branch and return 0   [45]
                         50     D6  0836  3920          INCL    R0                          ; If > 0, return 1              [45]
                                05  0838  3921 10$:     RSB                                 ;                               [45]
                                    0839  3922
                                    0839  3923 ;+
                                    0839  3924 ; IF_ADAPTER Action routine.
                                    0839  3925 ;
                                    0839  3926 ;       Returns:
                                    0839  3927 ;               1 => A adapter is present
                                    0839  3928 ;               0 => No adapter present
                                    0839  3929 ;-
                                    0839  3930 ACT_IF_ADAPTER:                             ;                               [48]
                          50    D4  0839  3931          CLRL    R0                          ; Assume no adapter present     [48]
```

```
                00000000'EF   D5  083B  3932          TSTL    L^q_adapter             ; See if the char-count is > zero       [48]
                          02  13  0841  3933          BEQL    10$                     ; If = 0, branch and return 0           [48]
                          50  D6  0843  3934          INCL    R0                      ; If > 0, return 1                      [48]
                              05  0845  3935 10$:     RSB                             ;                                       [48]
```

```
                        0846  3937 ;+
                        0846  3938 ; IF_REG_OR_ADR Routine.
                        0846  3939 ;
                        0846  3940 ;       Returns:
                        0846  3941 ;               1 => A register has been given OR an address has been given
                        0846  3942 ;               0 => No register AND no address were given
                        0846  3943 ;
                        0846  3944 ACT_IF_REG_OR_ADR:
                  50    D4    0846  3945         CLRL    R0                              ; Assume not set
              03 62     14    E1    0848  3946         BBC     #CLI$V_REG, -           ; If not REG, branch
                                    084C  3947                   CLI$L_FLAGS (R2), 20$
                  50    D6    084C  3948 10$:    INCL    R0                              ; One of them was set
                        05    084E  3949         RSB                                     ; Leave with 1
              F9 62     0B    E0    084F  3950 20$:    BBS     #CLI$V_ADR, -           ; If ADR, branch
                                    0853  3951                   CLI$L_FLAGS (R2), 10$
                        05    0853  3952         RSB                                     ; Leave with 0
                              0854  3953
                              0854  3954 ;+
                              0854  3955 ; IFNOTUSER action routine
                              0854  3956 ;
                              0854  3957 ;       Returns:
                              0854  3958 ;               1 => Not user mode
                              0854  3959 ;               0 => User mode
                              0854  3960 ;-
                              0854  3961 ACT_IFNOTUSER:                                ;                                       [38]
                  50    D4    0854  3962         CLRL    R0                              ; Pre-assume failure                    [38]
0000FE00'EF       1C    E0    0856  3963         BBS     #DSA$V_USER, -                  ;                                       [38]
                  02          085D  3964                   DSA$GL_FLAGS, 10$       ; Branch if not user mode               [38]
                  50    D6    085E  3965         INCL    R0                              ; Set R0 to cause branch                [38]
                        05    0860  3966 10$:    RSB                                     ;                                       [38]
```

```
                                0861  3968          .SUBTITLE       Xdelta Action Routines
                                0861  3969
                                0861  3970 ;+
                                0861  3971 ; VAX DEBUGGER ACTION
                                0861  3972 ;-
                                0861  3973
                                0861  3974          .WEAK   XDELBPT,XDELTBIT
                                0861  3975 ACT_IF_XDELTA:
      50   00000000'8F   D0    0861  3976          MOVL    #XDELBPT,R0             ; Is Xdelta linked?
                     08   13    0868  3977          BEQL    10$                     ; Branch if not
      0000FE00'EF    1C   E1    086A  3978          BBC     #DSA$V_USER, -
                     03         0871  3979                  DSA$GL_FLAGS, 20$       ; Branch if S/A
           50   01   D0    0872  3980 10$:    MOVL    #1,R0                   ; Make scanner take branch to error
                     05    0875  3981 20$:    RSB                             ; Return
                           0876  3982
                           0876  3983 ACT_XDELTA:
   04 A2   00000000'EF    9E    0876  3984          MOVAB   INI$BRK, -
                                087E  3985                  CLI$L_COMMAND(R2)       ; Set command address, Cause breakpoint
                     05    087E  3986          RSB                             ; Return to scanner
                           087F  3987
                     05    087F  3988          RSB                             ; RETURN FOR NEXT COMMAND
```

```
                                0880  3990          .SUBTITLE       Device, Brief, Adaptor Action Routines
                                0880  3991
                                0880  3992 ;+
                                0880  3993 ; SHOW DEVICE and SHOW DEVICE/BRIEF
                                0880  3994 ;-
                                0880  3995 ACT_DEVICE:
                09 62    1B     E0  0880  3996          BBS     #CLI$V_BRIEF,CLI$L_FLAGS(R2),5$ ; Check for brief flag.          [44]
                                0884  3997                                                                                    [44]
          04 A2 00000000'EF     9E  0884  3998          MOVAB   L^DSV$SHOWDEVICE,CLI$L_COMMAND(R2) ; SHOW DEV COMMAND ROUTINE  [44]
                                05  088C  3999          RSB                                        ;                          [44]
                                088D  4000                                                                                    [44]
          04 A2 00000000'EF     9E  088D  4001 5$:      MOVAB   L^DSV$SHOWDEVICEB,CLI$L_COMMAND(R2)     ; Show device brief  [44]
                                05  0895  4002          RSB                                        ;                          [44]
                                0896  4003                                                                                    [44]
                                0896  4004 ;+
                                0896  4005 ; /BRIEF Action routine.
                                0896  4006 ;-
                                0896  4007 ACT_BRIEF:                                ;                                       [44]
                00 62    1B     E2  0896  4008          BBSS    #CLI$V_BRIEF,CLI$L_FLAGS(R2),20$        ; Set flag          [44]
                         01     BB  089A  4009 20$:     PUSHR   #^M<R0>                 ; Save register                         [44]
             50 00000000'EF     9E  089C  4010          MOVAB   L^DSV$SHOWDEVICE,R0     ; Get address                           [44]
                04 A2    50     D1  08A3  4011          CMPL    R0,CLI$L_COMMAND(R2)    ; Check current command.                [44]
                         0A     12  08A7  4012          BNEQ    30$                     ; Skip if not show device.              [44]
          04 A2 00000000'EF     9E  08A9  4013          MOVAB   L^DSV$SHOWDEVICEB,CLI$L_COMMAND(R2) ; Change to BRIEF form.     [44]
                         15     11  08B1  4014          BRB     50$                     ;                                       [44]
             50 00000000'EF     9E  08B3  4015 30$:     MOVAB   L^DSV$SHOWSELECT,R0     ; Get address                           [44]
                04 A2    50     D1  08BA  4016          CMPL    R0,CLI$L_COMMAND(R2)    ; Check current command.                [44]
                         08     12  08BE  4017          BNEQ    50$                     ;                                       [44]
          04 A2 00000000'EF     9E  08C0  4018          MOVAB   L^DSV$SHOWSELECTB,CLI$L_COMMAND(R2)  ; Change to BRIEF form.  [44]
                         01     BA  08C8  4019 50$:     POPR    #^M<R0>                 ; Restore register                      [44]
                         05     08CA  4020          RSB                             ;                                       [44]
                                08CB  4021
                                08CB  4022 ;+
                                08CB  4023 ; BEGINADAPTER Action routine
                                08CB  4024 ; Adaptor name storage action routines for /ADAPTER qualifier
                                08CB  4025 ; on SELECT, DESELECT, and SHOW DEVICE commands.
                                08CB  4026 ;-
                                08CB  4027 ACT_BEGINADAPTER:
       00000004'EF    68        DE  08CB  4028          MOVAL   (R8), L^Q_ADAPTER+4     ; Save current pointer                  [43]
            00000000'EF        D4  08D2  4029          CLRL    L^Q_ADAPTER             ; Clear length field                    [43]
                         05     08D8  4030          RSB                             ; Return to caller                      [43]
                                08D9  4031
                                08D9  4032 ;+
                                08D9  4033 ; ADAPTER Action routine.
                                08D9  4034 ;-
                                08D9  4035 ACT_ADAPTER:
00000000'EF    58    00000004'EF     C3  08D9  4036          SUBL3   L^Q_ADAPTER+4, R8, -    ; Calculate count                [43]
                                08E5  4037                    L^Q_ADAPTER             ;                                       [43]
                         08     13  08E5  4038          BEQL    10$                     ;                                       [43]
          00 00000000 E2        18  08E7  4039          BBSS    #CLI$V_ADAPTER, -       ; Set adaptor bit                       [43]
                                08EF  4040                    L^CLI$L_FLAGS (R2), 10$ ;                                       [43]
                         05     08EF  4041 10$:     RSB                             ;                                       [43]
```

```
                                08F0  4043          .SUBTITLE       Show Select, Do_Cmd, Set Load, Show Load Action Routines
                                08F0  4044
                                08F0  4045 ;+
                                08F0  4046 ; SHOW SELECT/BRIEF and SHOW SELECT
                                08F0  4047 ;-
                                08F0  4048 ACT_SHOWSEL:
                09 62    1B  E0 08F0  4049          BBS     #CLI$V_BRIEF,CLI$L_FLAGS(R2),5$ ; Check for brief flags          [44]
                                08F4  4050
          04 A2 00000000'EF     9E 08F4  4051          MOVAB   L^DSV$SHOWSELECT,CLI$L_COMMAND(R2) ; SET COMMAND ROUTINE          [41]
                                05 08FC  4052          RSB                                                              [44]
                                08FD  4053
          04 A2 00000000'EF     9E 08FD  4054 5$:      MOVAB   L^DSV$SHOWSELECTB,CLI$L_COMMAND(R2)     ; Show select brief      [44]
                                05 0905  4055          RSB                                                              [44]
                                0906  4056
                                0906  4057 ;+
                                0906  4058 ; DO_CoMmanD
                                0906  4059 ;-
                                0906  4060 ACT_DO_CMD:
             50 04 A2           D0 0906  4061          MOVL    CLI$L_COMMAND(R2), R0           ; Check value before dispatch    [46]
                             02 13 090A  4062          BEQL    10$                             ; Don't dispatch if 0            [46]
                             60 16 090C  4063          JSB     (R0)                            ; Perform the routine            [46]
                                05 090E  4064 10$:     RSB                                     ;                                [46]
                                090F  4065
                                090F  4066 ;+
                                090F  4067 ; SET LOAD COMMAND
                                090F  4068 ;-
                                090F  4069 ACT_SETLOAD:
          04 A2 00000000'EF     9E 090F  4070          MOVAB   DSV$SETLOAD,CLI$L_COMMAND(R2)   ; Set the processor command
                                05 0917  4071          RSB
                                0918  4072
                                0918  4073 ;+
                                0918  4074 ; SHOW LOAD DEFAULTS
                                0918  4075 ;-
                                0918  4076 ACT_SHOWLOAD:
          04 A2 00000000'EF     9E 0918  4077          MOVAB   DSV$SHOWLOAD,CLI$L_COMMAND(R2)  ; Set the processor address
                                05 0920  4078          RSB
```

```
                        0921  4080          .SUBTITLE       MM On, Off, and Show Action Routines
                        0921  4081
                        0921  4082 ;+
                        0921  4083 ; SET MEMORY MANAGEMENT ON ROUTINE
                        0921  4084 ;-
                        0921  4085 ACT_MMON:
    04 A2    27'AF   9E 0921  4086          MOVAB   B^5$,CLI$L_COMMAND(R2); MEMORY ON ROUTINE
                     05 0926  4087          RSB
                        0927  4088
                        0927  4089 5$:      Br_If_Not_User  10$             ; Branch around if not user mode        [55]
                                                                                                                    [41]
  000000AB'EF   9F 092F  4090          PUSHAB  L^T_EMESSG1                     ;
            01  DD 0935  4091          pushl   #ds$k_printf                    ; Print with PRINTF                 [48]
            15  DD 0937  4092          pushl   #ds$k_type_command_err          ; Command error code                [48]
00000000'GF 03  FB 0939  4093          CALLS   #3, G^DSX$PRINT                 ; Print                             [48]
                     05 0940  4094          RSB
                        0941  4095
00000000'EF 01  D0 0941  4096 10$:      MOVL    #1,DS$GB_MM_ENB                 ; Flag MM ENABLED
00000000'EF 00  FB 0948  4097          CALLS   #0,DSX$MMON
                     05 094F  4098          RSB
                        0950  4099
                        0950  4100
                        0950  4101 ;+
                        0950  4102 ; SET MEMORY MANAGEMENT OFF ROUTINE
                        0950  4103 ;-
                        0950  4104 ACT_MMOFF:
    04 A2    56'AF   9E 0950  4105          MOVAB   B^5$,CLI$L_COMMAND(R2); MEMORY OFF ROUTINE
                     05 0955  4106          RSB
                        0956  4107
                        0956  4108 5$:      Br_If_Not_User  10$             ; Branch around if not in user mode     [55]
                                                                                                                    [41]
  000000AB'EF   9F 095E  4109          PUSHAB  L^T_EMESSG1                     ;
            01  DD 0964  4110          pushl   #ds$k_printf                    ; Print with PRINTF                 [48]
            15  DD 0966  4111          pushl   #ds$k_type_command_err          ; Command error code                [48]
00000000'GF 03  FB 0968  4112          CALLS   #3, G^DSX$PRINT                 ; Print                             [48]
                     05 096F  4113          RSB
                        0970  4114
  00000000'EF   94 0970  4115 10$:      CLRB    DS$GB_MM_ENB                    ; Allow it to turn MM OFF
00000000'EF 00  FB 0976  4116          CALLS   #0,DSX$MMOFF
                     05 097D  4117          RSB
                        097E  4118
                        097E  4119 ;+
                        097E  4120 ; SHOW STATUS OF MEMORY MANAGEMENT
                        097E  4121 ;-
                        097E  4122 ACT_SHOWMM:
    04 A2    84'AF   9E 097E  4123          MOVAB   B^10$,CLI$L_COMMAND(R2)         ; Set command processor
                     05 0983  4124          RSB
                        0984  4125
                        0984  4126 10$:      Br_If_Not_User  20$             ; Branch around if not in user mode     [55]
                        098C  4127          $PRINT  #ds$k_type_command_err, -        ; Print command error          [48]
                        098C  4128                    #ds$k_printf, -            ; .. use PRINTF                [48]
                        098C  4129                    L^T_EMESSG1                ; Can't do                     [48]
                     05 099F  4130          RSB
                        09A0  4131
  000000EF'EF   9F 09A0  4132 20$:      PUSHAB  T_OFF                           ; Assume MM OFF
  00000000'EF   95 09A6  4133          TSTB    DS$GB_MM_ENB                    ; is mm on?
            07  13 09AC  4134          BEQL    30$                             ; Branch if not
6E  000000EC'EF   9E 09AE  4135          MOVAB   T_ON,(SP)                       ; Change insert
                     09B5  4136
```

```
                00000DD0'EF   9F  09B5  4137 30$:    PUSHAB  T_SHOWMM                        ; Address of text
                          01  DD  09BB  4138         pushl   #ds$k_printf                   ; Print with PRINTF          [48]
                          16  DD  09BD  4139         pushl   #ds$k_type_command_out         ; Command output code       [48]
            00000000'GF   04  FB  09BF  4140         CALLS   #4, G^DSX$PRINT                ; Print                      [48]
                          05  09C6  4141         RSB                                    ; All done
```

```
                        09C7  4143          .SUBTITLE       Show Status, Show Support
                        09C7  4144
                        09C7  4145 ;+
                        09C7  4146 ; SHOW STATUS
                        09C7  4147 ;-
                        09C7  4148 ACT_SHOWSTATUS:                                                  ;               [39]
04 A2  00000000'EF   9E 09C7  4149          MOVAB   VRSHOWSTATUS, CLI$L_COMMAND(R2)                 ;               [39]
                     05 09CF  4150          RSB
                        09D0  4151
                        09D0  4152 ;+
                        09D0  4153 ; SHOW SUPPORT
                        09D0  4154 ;-
```

```
                                        09D0  4156 ACT_SHOWSUP:                                          ;                           [44]
         04 A2   00000000'EF      9E    09D0  4157         MOVAB   L^VRSUPPORT,CLI$L_COMMAND(R2)      ; Set command routine       [44]
                                  05    09D8  4158         RSB
                                        09D9  4159
                                        09D9  4160 ;+                                                              ;                     [46]
                                        09D9  4161 ; SHOW SECTIONS                                                 ;                     [46]
                                        09D9  4162 ;-                                                              ;                     [46]
                                        09D9  4163 ACT_SHOWSECTIONS:                                               ;                     [46]
         04 A2   00000000'EF      9E    09D9  4164         MOVAB   L^VRSHOWSECTIONS,CLI$._COMMAND(R2)      ; Set command routine [46]
                                  05    09E1  4165         RSB                                                     ;                     [46]
                                        09E2  4166
                                        09E2  4167 ;+
                                        09E2  4168 ; SHOW MEMORY
                                        09E2  4169 ;-
                                        C9E2  4170 Act_ShowMem:                                                    ;                     [51]
         04 A2   00000000'EF      9E    09E2  4171         MovAB   L^DSV$ShowMemory, Cli$L_Command(R2)     ; Show Memory address [51]
                                  05    09EA  4172         Rsb                                                     ;                     [51]
                                        09EB  4173     .Enable LSB                                                 ;                     [51]
                                        09EB  4174 /ct_ShoMemMap:                                                  ; /Map                [51]
                          50      D4    09EB  4175         ClrL    R0                                              ;                     [51]
                          0D      11    09ED  4176         BrB     10$                                             ;                     [51]
                                        09EF  4177 Act_ShoMemBuf:                                                  ; /Buffer             [51]
                   50     01      D0    09EF  4178         MovL    #1, R0                                          ;                     [51]
                          08      11    09F2  4179         BrB     10$                                             ;                     [51]
                                        09F4  4180 Act_ShoMemDat:                                                  ; /Data_Structure     [51]
                   50     02      D0    09F4  4181         MovL    #2, R0                                          ;                     [51]
                          03      11    09F7  4182         BrB     10$                                             ;                     [51]
                                        09F9  4183 Act_ShoMemAll:                                                  ; /All                [51]
                   50     03      D0    09F9  4184         MovL    #3, R0                                          ;                     [51]
    00 00000000'EF        50      E2    09FC  4185 10$:    BbsS    R0, L^ShowMemoryFlags, 20$                      ; set the bit         [51]
                                  05    0A04  4186 20$:    Rsb                                                     ; and return          [51]
                                        0A05  4187     .Disable LSB
                                        0A05  4188
                                        0A05  4189 ;+
                                        0A05  4190 ; SET MEMORY
                                        0A05  4191 ;-
                                        0A05  4192
                                        0A05  4193 ACT_SETMEM:                                                     ;                     [75]
         04 A2   00000A0E'EF      9F    0A05  4194         MOVAB   5$, CLI$L_COMMAND(R2)                           ; Address of following [75]
                                  05    0A0D  4195         RSB                                                     ; routine             [75]
                                        0A0E  4196
                                        0A0E  4197 5$:     BR_IF_NOT_USER  10$                                     ;                     [75]
                                        0A16  4198
              000000AB'EF         9F    0A16  4199         PUSHAB  L^T_EMESSG1                                     ; Not allowed in user mode[7
                          01      DD    0A1C  4200         PUSHL   #DS$K_PRINTF                                    ; Tell user           [75]
                          15      DD    0A1E  4201         PUSHL   #DS$K_TYPE_COMMAND_ERR                          ;                     [75]
          00000000'GF     03      FB    0A20  4202         CALLS   #3, G^DSX$PRINT                                 ;                     [75]
                                  05    0A27  4203         RSB                                                     ;                     [75]
                                        0A28  4204
                                        0A28  4205 10$:                                                            ;                     [75]
              00000000'EF         16    0A28  4206         JSB     L^VRSETMEM                                      ; Routine to do work  [75]
                                  05    0A2E  4207         RSB                                                     ;                     [75]
                                        0A2F  4208
                                        0A2F  4209         .END
```

$$$	= 00000080		D		ACT_IFNOTUSER	00000854	R	D	04
$$N	= 00000001		D		ACT_IF_ADAPTER	00000839	R	D	04
$ER	= 00000002		D		ACT_IF_FILE_SPEC	0000082F	R	D	04
$MODULE	00000000	R	D	03	ACT_IF_REG_OR_ADR	00000846	R	D	04
ABORT	= 00000006		D		ACT_IF_REQUIRED	00000826	R	D	04
ACT_ABORT	000002A7	R	D	04	ACT_IF_RUN	00000818	R	D	04
ACT_ADAPTER	000008D9	R	D	04	ACT_IF_XDELTA	00000861	R	D	04
ACT_ADDRESS	000005C9	R	D	04	ACT_ILC_CMD	0000025B	R	D	04
ACT_ADVANCE	000007F0	R	D	04	ACT_INC_CMD	00000261	R	D	04
ACT_ALL	000005FD	R	D	04	ACT_INITPCS	00000397	R	D	04
ACT_ASCII	000007B2	R	D	04	ACT_LAST	0000055E	R	D	04
ACT_ATTACH	00000296	R	D	04	ACT_LOAD	000003BF	R	D	04
ACT_BACKUP	000007E8	R	D	04	ACT_LONG	000007FA	R	D	04
ACT_BASE	000005B3	R	D	04	ACT_LOOP	000006CB	R	D	04
ACT_BEGINADAPTER	0000C8C8	R	D	04	ACT_MMOFF	00000950	R	D	04
ACT_BELL	000006A1	R	D	04	ACT_MMON	00000921	R	D	04
ACT_BINARY	000006A7	R	D	04	ACT_NEXT	000007AD	R	D	04
ACT_BREAK	000005D6	R	D	04	ACT_NEXT_INST	000003D2	R	D	04
ACT_BRIEF	00000896	R	D	04	ACT_NOTNOF	0000028C	R	D	04
ACT_BYTE	00000804	R	D	04	ACT_NULL	0000025A	R	D	04
ACT_CLEAR	00000280	R	D	04	ACT_OCT	00000B13	R	D	04
ACT_CLEARENFORCE	00000567	R	D	04	ACT_OPER	000006D1	R	D	04
ACT_CLEAR_CRD_DEBUG	00000699	R	D	04	ACT_OPTIONAL	0000052A	R	D	04
ACT_CLEAR_CRD_TRACE	00000654	R	D	04	ACT_PAGE	000006F5	R	D	04
ACT_CONT	000002E9	RG	D	04	ACT_PASS	00000540	R	D	04
ACT_CONTEXT	00000265	R	D	04	ACT_PREGN	000007E4	R	D	04
ACT_CONTINUE	000002BD	R	D	04	ACT_PROMPT	000006D7	R	D	04
ACT_CRD	00000449	R	D	04	ACT_QA	00000545	R	D	04
ACT_DATA	000007F5	R	D	04	ACT_QACL	00000737	R	D	04
ACT_DEATTACH	000002EA	R	D	04	ACT_QADEF	0000076B	R	D	04
ACT_DEBUG	000002F3	R	D	04	ACT_QAEP	00000731	R	D	04
ACT_DEBUG_SWITCH	00000534	R	D	04	ACT_QAMP	00000749	R	D	04
ACT_DEC	0000080E	R	D	04	ACT_QASL	00000743	R	D	04
ACT_DEFAULT	0000061E	R	D	04	ACT_QATL	0000073D	R	D	04
ACT_DEFAULT_DBG	000002FC	RG	D	04	ACT_QUICK	000006DD	R	D	04
ACT_DEPOSIT	00000316	R	D	04	ACT_REG12	000007BC	R	D	04
ACT_DESELECT	00000323	R	D	04	ACT_REG13	000007C2	R	D	04
ACT_DEVICE	00000880	R	D	04	ACT_REG14	000007C8	R	D	04
ACT_DIRECTORY	0000032C	R	D	04	ACT_REG15	000007D1	R	D	04
ACT_DIR_WIDE	00000338	R	D	04	ACT_REG16	000007D7	R	D	04
ACT_DO_CMD	00000906	R	D	04	ACT_REGM	000007CC	R	D	04
ACT_EFN	0000078D	R	D	04	ACT_REGN	00000787	R	D	04
ACT_ENUF	00000291	R	D	04	ACT_REGNUF	000007DF	R	D	04
ACT_EVENT	00000575	R	D	04	ACT_REGP	000007DB	R	D	04
ACT_EXAMINE	00000343	R	D	04	ACT_REQUIRED	0000052F	R	D	04
ACT_EXIT	00000350	R	D	04	ACT_RUN	000003E4	R	D	04
ACT_FAILURE	00000283	R	D	04	ACT_SCRIPT	00000437	R	D	04
ACT_FILEND	00000523	R	D	04	ACT_SEARCH	000006E3	R	D	04
ACT_FILESPEC	00000518	R	D	04	ACT_SECTION	0000054A	R	D	04
ACT_FLAGS	0000059E	R	D	04	ACT_SELECT	00000400	R	D	04
ACT_HALT	000006AD	R	D	04	ACT_SET	00000409	R	D	04
ACT_HELP	00000359	R	D	04	ACT_SETENFORCE	00000563	R	D	04
ACT_HEX	00000809	R	D	04	ACT_SETLOAD	0000090F	R	D	04
ACT_IE1	000006B3	R	D	04	ACT_SETMEM	00000A05	R	D	04
ACT_IE2	000006B9	R	D	04	ACT_SET_CRD_DEBUG	00000690	R	D	04
ACT_IE3	000006BF	R	D	04	ACT_SET_CRD_TRACE	0000064A	R	D	04
ACT_IE5	000006C5	R	D	04	ACT_SHOMEMACL	000009F9	R	D	04

```
ACT_SHOMEMBUF                   000009EF  R   D   04      CLI$K_CALL                      =  00000092     D
ACT_SHOMEMDAT                   000009F4  R   D   04      CLI$K_COMMA                     =  0000008E     D
ACT_SHOMEMMAP                   000009EB  R   D   04      CLI$K_DEC                       =  0000008A     D
ACT_SHOW                        00000416  R   D   04      CLI$K_EOL                       =  00000091     D
ACT_SHOWLOAD                    00000918  R   D   04      CLI$K_ERROR                     =  00000080     D
ACT_SHOWMEM                     000009E2  R   D   04      CLI$K_EXIT                      =  00000081     D
ACT_SHOWMM                      0000097E  R   D   04      CLI$K_FILE                      =  00000095     D
ACT_SHOWSECTIONS                000009D9  R   D   04      CLI$K_HEX                       =  00000089     D
ACT_SHOWSEL                     000008F0  R   D   04      CLI$K_KEYWORD                   =  0000008C     D
ACT_SHOWSTATUS                  000009C7  R   D   04      CLI$K_NUM                       =  00000085     D
ACT_SHOWSUP                     000009D0  R   D   04      CLI$K_OCT                       =  00000088     D
ACT_SHOW_CALLS                  0000063D  R   D   04      CLI$K_RETURN                    =  00000093     D
ACT_SHOW_CRD_TRACE              0000065E  R   D   04      CLI$K_SIZE                         00000444     D
ACT_START                       0000041B  R   D   04      CLI$K_SLASH                     =  00000090     D
ACT_SUBTEST                     0000C554  R   D   04      CLI$K_SPACE                     =  00000084     D
ACT_SUCCESS                     0000027A  R   D   04      CLI$K_STRING                    =  0000008B     D
ACT_SUMMARY                     00000440  R   D   04      CLI$K_SYMBOL                    =  0000008D     D
ACT_TEST                        00000559  R   D   04      CLI$K_VALUE                     =  0000008F     D
ACT_TRACE                       0G0006E9  R   D   04      CLI$L_ADDRESS                      00000018     D
ACT_VERIFY                      000006EF  R   D   04      CLI$L_COMMAND                      00000004     D
ACT_WIDTH                       00000713  R   D   04      CLI$L_DATA                         0000001C     D
ACT_WORD                        000007FF  R   D   04      CLI$L_FLAGS                        00000000     D
ACT_XDELTA                      00000876  R   D   04      CLI$L_LAST                         00000024     D
ADAPTER                      =  0000006F      D           CLI$L_NEXT                         00000030     D
ADDRESS                      =  00000037      D           CLI$L_PASS                         0000002C     D
ADVANCE                      =  0000005C      D           CLI$L_SUBT                         00000028     D
ALL                          =  00000039      D           CLI$L_TEST                         00000020     D
APT_COM                         ********    X   04      CLI$Q_BUFQWD                       00000034     D
APT_MSG                         ********    X   04      CLI$Q_FILE                         00000008     D
ASCII                        =  00000053      D           CLI$Q_SECTION                      00000010     D
ATTACH                       =  00000007      D           CLI$Q_TIME                         0000043C     D
BACKUP                       =  0000005B      D           CLI$T_BUFFER                       0000003C     D
BAD_LIST                        0000037D  R   D   03      CLI$V_ADAPTER                   =  00000018     D
BAD_QUALIFIER                   0000039D  R   D   03      CLI$V_ADR                       =  0000000B     D
BASE                         =  00000036      D           CLI$V_ASCII                     =  00000013     D
BEGIN                           ********    X   04      CLI$V_BREAK                     =  0000000A     D
BEGINADAPTER                 =  0000006E      D           CLI$V_BRIEF                     =  0000001B     D
BELL                         =  00000038      D           CLI$V_BYTE                      =  0000000D     D
BINARY                       =  0000003C      D           CLI$V_CLEAR                     =  00000002     D
BIT...                       =  00000003      D           CLI$V_DEC                       =  00000010     D
BREAK                        =  00000038      D           CLI$V_DEFAULT                   =  0000000C     D
BRIEF                        =  0000006D      D           CLI$V_DEPOSIT                   =  00000019     D
BYTE                         =  00000060      D           CLI$V_EVENT                     =  00000008     D
B_SUCCESS                       0000044C  R   D   02      CLI$V_EXAM                      =  00000005     D
CLEAR                        =  00000008      D           CLI$V_FLAGS                     =  00000009     D
CLEARENFORCE                 =  00000033      D           CLI$V_HEX                       =  00000012     D
CLEAR_CRD_DEBUG              =  0000001E      D           CLI$V_KERNEL                    =  00000017     D
CLEAR_CRD_TRACE              =  00000018      D           CLI$V_LOAD                      =  00000006     D
CLEAR_FLAGS_LOOP                000009AE  R   D   03      CLI$V_LONG                      =  0000000F     D
CLEAR_PARAMS                    000008D6  R   D   03      CLI$V_NOTNUF                    =  00000001     D
CLI                             00000072  R   D   04      CLI$V_OCT                       =  00000011     D
CLI$K_ALNUM                  =  00000087      D           CLI$V_PREG                      =  0000001A     D
CLI$K_ALPHA                  =  00000086      D           CLI$V_QA                        =  00000007     D
CLI$K_BIF                    =  00000083      D           CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$K_BIFS                   =  00000094      D           CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$K_BR                     =  00000082      D           CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$K_BUFSIZ                 =  00000100      D           CLI$V_QASUBTESTLOOPS            =  0000001E     D
```

```
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00C00000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
CLI_ACTION                         0000013D RG  D  04
COMMAND_TREE                       00000180 RG  D  03
COMMON_CRD_TRACE_DEBUG             00000689 R   D  04
COMMON_ENFORCE                     0000056A R   D  04
CONTEXT                         =  00000003     D
CONTINUE                        =  00000009     D
CRD                             =  0000C023     D
CRD$GL_CRD_COMMAND_DEBUG           ********   X    04
CRD$K_CRD_INITIALIZATION        =  00000000   G D
CRD$K_DS_CONTROL_C              =  00000001   G D
CRD$K_DS_FLAGS                  =  00000000   G D
CRD$K_FAILURE                   =  00000000     D
CRD$K_HOOK_POINT                =  00000000   G D
CRD$K_LAST_ACCESS_CODE          =  00000002   G D
CRD$K_LAST_DS_VARIABLE          =  00000002   G D
CRD$K_LAST_FUNCTION             =  00000002   G D
CRD$K_LAST_HOOK_POINT           =  00000001   G D
CRD$K_READ                      =  00000000   G D
CRD$K_SUCCESS                   =  00000001     D
CRD$K_TYPECODE                  =  00000001   G D
CRD$K_WRITE                     =  00000001   G D
DATA                            =  0000005D     D
DBG_NAME                           0000030D R   D  04
DEATTACH                        =  0000000A     D
DEATTACH_QUAL                      000003E9 R   D  03
DEBUG                           =  0000000B     D
DEBUG$GB_FLAGS                     ********   X    04
DEBUG_SWITCH                    =  0000000C     D
DEC                             =  00000062     D
DEFAULT                         =  0000003A     D
DEFAULT_DBG                     =  0000000D     D
DEPOSIT                         =  0000000E     D
DEPOSIT_PARAMS                     00000E76 R   D  03
DESELECT                        =  00000070     D
DESELECT_QUALS                     00000435 R   D  03
DEVICE                          =  0000006C     D
DIRECTORY                       =  0000000F     D
DIR_WIDE                        =  00000080     D
DO_CMD                          =  00000073     D
DS$A_PRGBGN                        ********   X    04
DS$CLI                             00000000 RG  D  04
DS$GA_CRD_DS_INTERFACE             ********   X    04
DS$GA_DS_CTRL_C_FIRST              ********   X    04
DS$GA_DS_CTRL_C_SECOND             ********   X    04
DS$GB_INHIBIT_NAMING               ********   X    04
DS$GB_MM_ENB                       ********   X    04
DS$GL_CLIBASE                      00000000 RG  D  02
DS$GL_CRD_DEBUG                    ********   X    04
DS$GL_FLAGS                        ********   X    04
DS$GT_PROMPT                       ********   X    04
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_PRINTB                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  00000018     D
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        =  0000000B     D
DS$K_TYPE_FIRST_PASS            =  00000011     D
DS$K_TYPE_GENERAL               =  00000000     D
DS$K_TYPE_GENERAL_ERROR         =  00000003     D
DS$K_TYPE_NO_TESTS              =  00000012     D
DS$K_TYPE_PARAM_ERROR           =  0000001C     D
DS$K_TYPE_PROGRAM_END           =  00000010     D
DS$K_TYPE_PROGRAM_INFO          =  00000017     D
DS$K_TYPE_PROGRAM_START         =  0000000F     D
DS$K_TYPE_QIO_INVADP            =  00000024     D
DS$K_TYPE_QIO_NODRIVER          =  00000022     D
DS$K_TYPE_QIO_WRONGVER          =  00000023     D
DS$K_TYPE_SCRIPT_ECHO           =  00000021     D
DS$K_TYPE_SCRIPT_PNF            =  0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         =  00000020     D
DS$K_TYPE_SCRIPT_SKIP           =  0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        =  00000019     D
DS$K_TYPE_START_ERR             =  00000018     D
DS$K_TYPE_START_LIST            =  00000025     D
DS$K_TYPE_SUMMARY               =  0000000E     D
DS$K_TYPE_USER_PROMPT           =  00000002     D
DS$K_WARNING                    =  00000000     D
DS$L_USERCNTRLC                    ********   X    04
DS$M_ABRTFLG                    =  00000040     D
DS$M_BADTIME                    =  00100000     D
DS$M_BATCH                      =  00400000     D
DS$M_BRKCLR                     =  00001000     D
DS$M_BRKPT                      =  00000800     D
DS$M_CHARFLG                    =  00000100     D
DS$M_CMDFLG                     =  00000080     D
DS$M_CTRLC                      =  00000001     D
```

DS$M_CTRLO	= 00010000	D	DS$_IHWE	= 00660060	D		
DS$M_DEVFLG	= 00000200	D	DS$_ILLCHAR	= 00660018	D		
DS$M_DISABLCC	= 01000000	D	DS$_ILLPAGCNT	= 00660078	D		
DS$M_DONFLG	= 00002000	D	DS$_ILLUNIT	= 00660100	D		
DS$M_ERRFLG	= 00000010	D	DS$_INSFMEM	= 00660050	D		
DS$M_EXCEPT	= 00080000	D	DS$_IPL2HI	= 006600B8	D		
DS$M_EXETST	= 00040000	D	DS$_IVADDR	= 00660040	D		
DS$M_HLTFLG	= 00000008	D	DS$_IVVECT	= 00660038	D		
DS$M_LODFLG	= 00000002	D	DS$_KRNLSTK	= 00660090	D		
DS$M_MEMMGT	= 00008000	D	DS$_LOGIC	= 00660070	D		
DS$M_OUTPUT	= 00800000	D	DS$_MCHK	= 00660088	D		
DS$M_RUBFLG	= 00000020	D	DS$_MMOFF	= 00660058	D		
DS$M_SCRIPT	= 00200000	D	DS$_NEEDUNIT	= 006600F8	D		
DS$M_SETIMR	= 02000000	D	DS$_NODE	= 00660128	D		
DS$M_STRFLG	= 00000004	D	DS$_NOPCS	= 00660110	D		
DS$M_SUBT	= 00004000	D	DS$_NORMAL	= 00660001	D		
DS$M_SYSFLG	= 00000400	D	DS$_NOSUPPORT	= 006600B1	D		
DS$M_TIMRON	= 00020000	D	DS$_NOTDON	= 00660030	D		
DS$V_ABRTFLG	= 00000006	D	DS$_NOTIMP	= 006600B0	D		
DS$V_BADTIME	= 00000014	D	DS$_NULLSTR	= 00660010	D		
DS$V_BATCH	= 00000016	D	DS$_OVERFLOW	= 00660008	D		
DS$V_BRKCLR	= 0000000C	D	DS$_POWER	= 00660098	D		
DS$V_BRKPT	= 0000000B	D	DS$_PROGERR	= 00660020	D		
DS$V_CHARFLG	= 00000008	D	DS$_SEVERE	= 00660004	D		
DS$V_CMDFLG	= 00000007	D	DS$_TRANSL	= 006600A0	D		
DS$V_CTRLC	= 00000000	D	DS$_TRUNCATE	= 00660028	D		
DS$V_CTRLO	= 00000010	D	DS$_UNEXPINT	= 006600D8	D		
DS$V_DEVFLG	= 00000009	D	DS$_VASFULL	= 00660048	D		
DS$V_DISABLCC	= 00000018	D	DS$_WARNING	= 00660000	D		
DS$V_DONFLG	= 0000000D	D	DSA$AL_APTMAIL	0000FE00	D		
DS$V_ERRFLG	= 00000004	D	DSA$AT_APTTXT	0000FA00	D		
DS$V_EXCEPT	= 00000013	D	DSA$GL_APTCOM	0000FE04	D		
DS$V_EXETST	= 00000012	D	DSA$GL_DEVLEN	0000FE58	D		
DS$V_HLTFLG	= 00000003	D	DSA$GL_ERRNO	0000FE44	D		
DS$V_LODFLG	= 00000001	D	DSA$GL_EVENT	0000FE48	D		
DS$V_MEMMGT	= 0000000F	D	DSA$GL_FLAGS	0000FE00	D		
DS$V_OUTPUT	= 00000017	D	DSA$GL_MSGTYP	0000FE40	D		
DS$V_RUBFLG	= 00000005	D	DSA$GL_PASSES	0000FE08	D		
DS$V_SCRIPT	= 00000015	D	DSA$GL_PASSNO	0000FE54	D		
DS$V_SETIMR	= 00000019	D	DSA$GL_SECTNO	0000FE10	D		
DS$V_STRFLG	= 00000002	D	DSA$GL_SID	0000FE14	D		
DS$V_SUBT	= 0000000E	D	DSA$GL_SUBTNO	0000FE4C	D		
DS$V_SYSFLG	= 0000000A	D	DSA$GL_TESTNO	0000FE50	D		
DS$V_TIMRON	= 00000011	D	DSA$GL_UNITS	0000FE0C	D		
DS$_ARITH	= 006600D0	D	DSA$GQ_MSGPTR	0000FE68	D		
DS$_ASBE	= 00660118	D	DSA$GT_DEVNAM	0000FE5C	D		
DS$_BADLINK	= 006600F0	D	DSA$V_APT	= 0000001F	D		
DS$_BADTYPE	= 006600E8	D	DSA$V_BELL	= 00000003	D		
DS$_BIIC	= 00660120	D	DSA$V_BINARY	= 00000001	D		
DS$_CHME	= 006600A8	D	DSA$V_CRD_AUTOTEST_OFF	= 0000000B	D		
DS$_CHMK	= 006600E0	D	DSA$V_CRD_MENUTEST_OFF	= 00000010	D		
DS$_DEVNAME	= 00660108	D	DSA$V_CRD_MENUTEST_ON	= 00000012	D		
DS$_ERROR	= 00660002	D	DSA$V_CRD_TRACE	= 00000011	D		
DS$_FHWE	= 00660068	D	DSA$V_HALT	= 00000000	D		
DS$_FRAGBUF	= 00660080	D	DSA$V_IE1	= 00000004	D		
DS$_ICBUSY	= 006600C8	D	DSA$V_IE2	= 00000005	D		
DS$_ICERR	= 006600C0	D	DSA$V_IE3	= 00000006	D		

```
DSA$V_IES                       =  00000007     D
DSA$V_LOAD_DEBUGGER             =  0000001E     D
DSA$V_LOOP                      =  00000002     D
DSA$V_OPER                      =  0000000C     D
DSA$V_PROMPT                    =  0000000D     D
DSA$V_QUICK                     =  00000008     D
DSA$V_SEARCH                    =  0000000E     D
DSA$V_TRACE                     =  0000000A     D
DSA$V_USER                      =  0000001C     D
DSA$V_VERIFY                    =  00000009     D
DSR$COMPLETION                     ********   X   04
DSR$LOAD_CRD                       ********   X   04
DSR$UNLOAD_CRD                     ********   X   04
DSV$ATTACH                         ********   X   04
DSV$DEATTACH                       ********   X   04
DSV$DEBUG                          ********   X   04
DSV$DESELECT                       ********   X   04
DSV$DIRECTORY                      ********   X   04
DSV$EXIT                           ********   X   04
DSV$INITPCS                        ********   X   04
DSV$LOAD                           ********   X   04
DSV$RUN                            ********   X   04
DSV$SELECT                         ********   X   04
DSV$SETLOAD                        ********   X   04
DSV$SETPAGE                        ********   X   04
DSV$SHOWDEVICE                     ********   X   04
DSV$SHOWDEVICEB                    ********   X   04
DSV$SHOWLOAD                       ********   X   04
DSV$SHOWMEMORY                     ********   X   04
DSV$SHOWPAGE                       ********   X   04
DSV$SHOWSELECT                     ********   X   04
DSV$SHOWSELECTB                    ********   X   04
DSX$HELP                           ********   X   04
DSX$MMOFF                          ********   X   04
DSX$MMON                           ********   X   04
DSX$PRINT                          ********   X   04
DSX$SUPERPARSE                     ********   X   04
DS_CLEANUP                         ********   X   04
DS_ERRSUP                          ********   X   04
DS_SUPERLINE                       ********   X   04
EFN                             =  0000004F     D
END_CLEAR_FLAGS_LOOP               00000A74 R   D   03
END_SET_FLAGS_LOOP                 00000D78 R   D   03
ENUF                            =  00000015     D
EOL                                00000349 R   D   03
EVENT                           =  00000034     D
EXAMINE                         =  00000010     D
EXAMINE_PARAMS                     00000D87 R   D   03
EXIT                            =  00000011     D
FAILURE                         =  00000005     D
FALSE                           =  00000000     D
FILEND                          =  00000025     D
FILESPEC                        =  00000024     D
FLAGS                           =  00000035     D
GET_DEV_NAME                       000003BD R   D   03
GET_FILE_SPEC                      0000045D R   D   03
GT_MENU_ERROR                      000000F3 RG  D   03
HALT                            =  0000003D     D
HELP                            =  00000012     D
HELPINFO                           ********   X   04
HEX                             =  00000061     D
IE1                             =  0000003E     D
IE2                             =  0000003F     D
IE3                             =  00000040     D
IES                             =  00000041     D
IFNOTUSER                       =  00000069     D
IF_ADAPTER                      =  00000067     D
IF_FILE_SPEC                    =  00000066     D
IF_REG_OR_ADR                   =  00000068     D
IF_REQUIRED                     =  00000065     D
IF_RUN                          =  00000064     D
IF_XDELTA                       =  0000006A     D
ILLEGAL                            00000361 R   D   03
ILL_CMD                         =  00000001     D
INCOMPLETE                         00000379 R   D   03
INC_CMD                         =  00000002     D
INI$BRK                            ********   X   04
INITPCS                         =  0000007F     D
INIT_CONTEXT                       ********   X   04
KB_CHECK                           ********   X   04
KB_CHECK_APT                       ********   X   04
LAST                            =  00000029     D
LINE_COUNT                         ********   X   04
LOAD                            =  00000013     D
LONG                            =  0000005E     D
LOOP                            =  00000042     D
MAPFREE                            ********   X   04
MMOFF                           =  0000002C     D
MMON                            =  00000028     D
NEXT                            =  00000050     D
NEXT_INST                       =  00000014     D
NOTNUF                          =  00000016     D
NULL                            =  00000000     D
OCT                             =  00000063     D
OFF                             =  00000000     D
ON                              =  00000001     D
OPER                            =  00000043     D
OPTIONAL                        =  00000026     D
PAGE                            =  00000051     D
PASS                            =  00000030     D
PREGN                           =  0000002E     D
PROMPT                          =  00000044     D
QA                              =  00000031     D
QACL                            =  0000004A     D
QADEF                           =  0000004E     D
QAEP                            =  00000049     D
QAMP                            =  0000004D     D
QASL                            =  0000004C     D
QATL                            =  0000004B     D
QA_COMMON                          0000074D R   D   04
QUICK                           =  00000045     D
Q_ADAPTER                          ********   X   04
Q_GOOD_CMD                         00000444 R   D   02
RCLI                               00000002 R   D   04
```

```
REG12                           = 00000055     D
REG13                           = 00000056     D
REG14                           = 00000057     D
REG15                           = 00000058     D
REG16                           = 00000059     D
REGN                            = 00000054     D
REGP                            = 0000005A     D
REQUIRED                        = 00000027     D
RUN                             = 00000017     D
SCRIPT                          = 00000018     D
SCRIPT$CONT                       ********   X    04
SCRIPT$FLUSH                      ********   X    04
SCRIPT$OPEN                       ********   X    04
SEARCH                          = 00000046     D
SECTION                         = 0000002F     D
SELECT                          = 00000071     D
SELECT_QUALS                      00000435 R   D    03
SET                             = 00000019     D
SETENFORCE                      = 00000032     D
SETLOAD                         = 00000074     D
SETMEM                          = 0000007E     D
SET_CRD_DEBUG                   = 0000001D     D
SET_CRD_TRACE                   = 0000001C     D
SET_DEFAULT_PARAMS                00000F5F R   D    03
SET_FLAGS_LOOP                    00000C9F R   D    03
SET_PARAMS                        00000A80 R   D    03
SHOMEMALL                       = 0000007D     D
SHOMEMBUF                       = 0000007B     D
SHOMEMDAT                       = 0000007C     D
SHOMEMMAP                       = 0000007A     D
SHOW                            = 0000001A     D
SHOWLOAD                        = 00000075     D
SHOWMEM                         = 00000079     D
SHOWMEMORYFLAGS                   ********   X    04
SHOWMM                          = 0000002D     D
SHOWSECTIONS                    = 00000078     D
SHOWSEL                         = 00000072     D
SHOWSTATUS                      = 00000077     D
SHOWSUP                         = 00000076     D
SHOW_CALLS                      = 00000020     D
SHOW_CRD_TRACE                  = 0000001F     D
SHOW_PARAMS                       00000648 R   D    03
SIZ...                          = 00000001     D
SS$_ENDOFFILE                     ********   X    04
START                           = 00000021     D
START_AND_RUN                     00000509 R   D    03
START_RUN_QUALIFIERS              000005C5 R   D    03
SUBTEST                         = 0000002A     D
SUCCESS                         = 00000004     D
SUMMARY                         = 00000022     D
SYS$EXIT                          ********  GX    04
TEST                            = 00000028     D
TRACE                           = 00000047     D
TRUE                            = 00000001     D
T_CONT                            00000092 R   D    03
T_CRD_TRACE_OUTPUT                00000163 R   D    03
T_EMESSG1                         000000AB RG  D    03
T_ILLCMD                          0000001B R   D    03
T_ILLEFN                          00000061 R   D    03
T_INCOMPLETE                      0000003A R   D    03
T_OFF                             000000EF R   D    03
T_ON                              000000EC R   D    03
T_PRGERR                          00000004 R   D    03
T_SHOWMM                          000000D0 R   D    03
T_USER_MODE_MENU_ERROR            00000129 R   D    03
VERIFY                          = 00000048     D
VRABORT                           ********   X    04
VRCLRBRK                          ********   X    04
VRCLREF                           ********   X    04
VRCLRFLG                          ********   X    04
VRDEPOSIT                         ********   X    04
VREXAMINE                         ********   X    04
VRSETBASE                         ********   X    04
VRSETBRK                          ********   X    04
VRSETDFLT                         ********   X    04
VRSETEF                           ********   X    04
VRSETFLG                          ********   X    04
VRSETMEM                          ********   X    04
VRSETQA                           ********   X    04
VRSETWIDTH                        ********   X    04
VRSHOWBASE                        ********   X    04
VRSHOWBRK                         ********   X    04
VRSHOWDFLT                        ********   X    04
VRSHOWEF                          ********   X    04
VRSHOWFLG                         ********   X    04
VRSHOWQA                          ********   X    04
VRSHOWSECTIONS                    ********   X    04
VRSHOWSTATUS                      ********   X    04
VRSHOWWIDTH                       ********   X    04
VRSHOW_CALLS                      ********   X    04
VRSTART                           ********   X    04
VRSUMMARY                         ********   X    04
VRSUPPORT                         ********   X    04
WIDE_FLAG                         ********   X    04
WIDTH                           = 00000052     D
WORD                            = 0000005F     D
XDELBPT                           ********W GX    04
XDELTA                          = 0000006B     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000044D (1101.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000FCC (4044.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000A2F (2607.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00000080	2738 (6)	#-2603 (6)
$$N	=00000001	4129 (10)	#-2480 (6) 2606 (6) 2607 (6) 2608 (6) 2609 (6) 2610 (6) 2611 (6) 2612 (6) 2613 (6) 2614 (6) 2615 (6) 2616 (6) 2617 (6) 2618 (6) 2619 (6) 2620 (6) 2621 (6) 2622 (6) 2623 (6) 2624 (6) 2625 (6) 2626 (6) 2627 (6) 2628 (6) 2629 (6) 2630 (6) 2631 (6) 2632 (6) 2633 (6) 2634 (6) 2635 (6) 2636 (6) 2637 (6) 2638 (6) 2639 (6) 2640 (6) 2641 (6) 2642 (6) 2643 (6) 2644 (6) 2645 (6) 2646 (6) 2647 (6) 2648 (6) 2649 (6) 2650 (6) 2651 (6) 2652 (6) 2653 (6) 2654 (6) 2655 (6) 2656 (6) 2657 (6) 2658 (6) 2659 (6) 2660 (6) 2661 (6) 2662 (6) 2663 (6) 2664 (6) 2665 (6) 2666 (6) 2667 (6) 2668 (6) 2669 (6) 2670 (6) 2671 (6) 2672 (6) 2673 (6) 2674 (6) 2675 (6) 2676 (6) 2677 (6) 2678 (6) 2679 (6) 2680 (6) 2681 (6) 2682 (6) 2683 (6) 2684 (6) 2685 (6) 2686 (6) 2687 (6) 2688 (6) 2689 (6) 2690 (6) 2691 (6) 2692 (6) 2693 (6) 2694 (6) 2695 (6) 2696 (6) 2697 (6) 2698 (6) 2699 (6) 2700 (6) 2701 (6) 2702 (6) 2703 (6) 2704 (6) 2705 (6) 2706 (6) 2707 (6) 2708 (6) 2709 (6) 2710 (6) 2711 (6) 2712 (6) 2713 (6) 2714 (6) 2715 (6) 2716 (6) 2717 (6) 2718 (6) 2719 (6) 2720 (6) 2721 (6) 2722 (6) 2723 (6) 2724 (6) 2725 (6) 2726 (6) 2727 (6) 2728 (6) 2729 (6) 2730 (6) 2731 (6) 2732 (6) 2733 (6) 2734 (6) 2738 (6) 2850 (7) #-3109 (7) #-3138 (7) 3470 (7) #-4129 (10)
$ER	=00000002	2740 (6)	#-2740 (6)
$MODULE	00000000-R	395 (1)	2740 (6)
ABORT	=00000006	2612 (6)	453 (1)
ACT_ABORT	000002A7-R	2824 (7)	2612 (6)
ACT_ADAPTER	000008D9-R	4035 (10)	2717 (6)
ACT_ADDRESS	000005C9-R	3364 (7)	2661 (6) #-3796 (10)
ACT_ADVANCE	000007F0-R	3812 (10)	2698 (6)
ACT_ALL	000005FD-R	3395 (7)	2663 (6)
ACT_ASCII	000007B2-R	3749 (10)	2689 (6)
ACT_ATTACH	00000296-R	2815 (7)	2613 (6)
ACT_BACKUP	000007E8-R	3803 (10)	2697 (6) #-3774 (10)
ACT_BASE	000005B3-R	3351 (7)	2660 (6)
ACT_BEGINADAPTER	000008CB-R	4027 (10)	2716 (6)
ACT_BELL	000006A1-R	3491 (7)	2665 (6)
ACT_BINARY	000006A7-R	3502 (10)	2666 (6)

```
ACT_BREAK                       000005D6-R  3373  (7)     2662  (6)
ACT_BRIEF                       00000896-R  4007  (10)    2715  (6)
ACT_BYTE                        00000804-R  3846  (10)    2702  (6)
ACT_CLEAR                       000002B0-R  2832  (7)     2614  (6)
ACT_CLEARENFORCE                00000567-R  3304  (7)     2657  (6)
ACT_CLEAR_CRD_DEBUG             00000699-R  3481  (7)     2636  (6)
ACT_CLEAR_CRD_TRACE             00000654-R  3454  (7)     2633  (6)    #-3484   (7)
ACT_CONT                        000002E9-R  2853  (7)     2984  (7)
ACT_CONTEXT                     00000265-R  2773  (7)     2609  (6)    #-2758   (7)    #-2759   (7)    #-2766   (7)
ACT_CONTINUE                    000002BD-R  2842  (7)     2615  (6)
ACT_CRD                         00000449-R  3062  (7)     2641  (6)
ACT_DATA                        000007F5-R  3821  (10)    2699  (6)
ACT_DEATTACH                    000002EA-R  2860  (7)     2616  (6)
ACT_DEBUG                       000002F3-R  2869  (7)     2617  (6)
ACT_DEBUG_SWITCH                00000534-R  3185  (7)     2618  (6)
ACT_DEC                         0000080E-R  3862  (10)    2704  (6)
ACT_DEFAULT                     0000061E-R  3412  (7)     2664  (6)
ACT_DEFAULT_DBG                 000002FC-R  2873  (7)     2619  (6)
ACT_DEPOSIT                     00000316-R  2883  (7)     2620  (6)
ACT_DESELECT                    00000323-R  2893  (7)     2718  (6)
ACT_DEVICE                      00000880-R  3995  (10)    2714  (6)
ACT_DIRECTORY                   0000032C-R  2901  (7)     2621  (6)
ACT_DIR_WIDE                    0000033B-R  2907  (7)     2734  (6)
ACT_DO_CMD                      00000906-R  4060  (10)    2721  (6)
ACT_EFN                         0000078D-R  3722  (10)    2685  (6)
ACT_ENUF                        00000291-R  2806  (7)     2627  (6)
ACT_EVENT                       00000575-R  3315  (7)     2658  (6)
ACT_EXAMINE                     00000343-R  2914  (7)     2622  (6)
ACT_EXIT                        00000350-R  2924  (7)     2623  (6)
ACT_FAILURE                     00000283-R  2791  (7)     2611  (6)
ACT_FILEND                      00000523-R  3159  (7)     2643  (6)
ACT_FILESPEC                    0000051B-R  3151  (7)     2642  (6)
ACT_FLAGS                       0000059E-R  3335  (7)     2659  (6)
ACT_HALT                        000006AD-R  3510  (10)    2667  (6)
ACT_HELP                        00000359-R  2934  (7)     2624  (6)
ACT_HEX                         00000809-R  3854  (10)    2703  (6)
ACT_IE1                         000006B3-R  3518  (10)    2668  (6)
ACT_IE2                         000006B9-R  3526  (10)    2669  (6)
ACT_IE3                         000006BF-R  3534  (10)    2670  (6)
ACT_IES                         000006C5-R  3542  (10)    2671  (6)
ACT_IFNOTUSER                   00000854-R  3961  (10)    2711  (6)
ACT_IF_ADAPTER                  00000839-R  3930  (10)    2709  (6)
ACT_IF_FILE_SPEC                0000082F-R  3916  (10)    2708  (6)
ACT_IF_REG_OR_ADR               00000846-R  3944  (10)    2710  (6)
ACT_IF_REQUIRED                 00000826-R  3902  (10)    2707  (6)
ACT_IF_RUN                      00000818-R  3886  (10)    2706  (6)
ACT_IF_XDELTA                   00000861-R  3975  (10)    2712  (6)
ACT_ILC_CMD                     0000025B-R  2755  (6)     2607  (6)
ACT_INC_CMD                     00000261-R  2764  (7)     2608  (6)
ACT_INITPCS                     00000397-R  2953  (7)     2733  (6)
ACT_LAST                        0000055E-R  3231  (7)     2647  (6)
ACT_LOAD                        000003BF-R  2970  (7)     2625  (6)
ACT_LONG                        000007FA-R  3830  (10)    2700  (6)
ACT_LOOP                        000006CB-R  3552  (10)    2672  (6)
ACT_MMOFF                       00000950-R  4104  (10)    2650  (6)
ACT_MMON                        00000921-R  4085  (10)    2649  (6)
ACT_NEXT                        000007AD-R  3742  (10)    2686  (6)
```

ACT_NEXT_INST	00000302-R	2981	(7)	2626	(6)		
ACT_NOTNUF	0000028C-R	2798	(7)	2628	(6)		
ACT_NULL	0000025A-R	2749	(6)	2606	(6)		
ACT_OCT	00000813-R	3870	(10)	2705	(6)		
ACT_OPER	000006D1-R	3560	(10)	2673	(6)		
ACT_OPTIONAL	0000052A-R	3167	(7)	2644	(6)		
ACT_PAGE	000006F5-R	3612	(10)	2687	(6)		
ACT_PASS	00000540-R	3193	(7)	2654	(6)		
ACT_PREGN	000007E4-R	3794	(10)	2652	(6)		
ACT_PROMPT	000006D7-R	3568	(10)	2674	(6)		
ACT_QA	00000545-R	3200	(7)	2655	(6)		
ACT_QACL	00000737-R	3654	(10)	2680	(6)		
ACT_QADEF	0000076B-R	3693	(10)	2684	(6)		
ACT_QAEP	00000731-R	3649	(10)	2679	(6)		
ACT_QAMP	00000749-R	3669	(10)	2683	(6)		
ACT_QASL	00000743-R	3664	(10)	2682	(6)		
ACT_QATL	0000073D-R	3659	(10)	2681	(6)		
ACT_QUICK	000006DD-R	3576	(10)	2675	(6)		
ACT_REG12	000007BC-R	3764	(10)	2691	(6)		
ACT_REG13	000007C2-R	3767	(10)	2692	(6)		
ACT_REG14	000007C8-R	3770	(10)	2693	(6)		
ACT_REG15	000007D1-R	3777	(10)	2694	(6)		
ACT_REG16	000007D7-R	3781	(10)	2695	(6)		
ACT_REGM	000007CC-R	3772	(10)	#-3766	(10)	#-3769	(10)
ACT_REGN	000007B7-R	3759	(10)	2690	(6)		
ACT_REGNUF	000007DF-R	3786	(10)	#-3785	(10)		
ACT_REGP	000007DB-R	3783	(10)	2696	(6)	#-3779	(10)
ACT_REQUIRED	0000052F-R	3175	(7)	2645	(6)		
ACT_RUN	000003E4-R	2990	(7)	2629	(6)		
ACT_SCRIPT	00000437-R	3043	(7)	2630	(6)		
ACT_SEARCH	000006E3-R	3586	(10)	2676	(6)		
ACT_SECTION	0000054A-R	3208	(7)	2653	(6)		
ACT_SELECT	00000400-R	3005	(7)	2719	(6)		
ACT_SET	00000409-R	3012	(7)	2631	(6)		
ACT_SETENFORCE	00000563-R	3301	(7)	2656	(6)		
ACT_SETLOAD	0000090F-R	4069	(10)	2722	(6)		
ACT_SETMEM	00000A05-R	4193	(11)	2732	(6)		
ACT_SET_CRD_DEBUG	00000690-R	3476	(7)	2635	(6)		
ACT_SET_CRD_TRACE	0000064A-R	3448	(7)	2634	(6)	#-3479	(7)
ACT_SHOMEMALL	000009F9-R	4183	(11)	2731	(6)		
ACT_SHOMEMBUF	000009EF-R	4177	(11)	2729	(6)		
ACT_SHOMEMDAT	000009F4-R	4180	(11)	2730	(6)		
ACT_SHOMEMMAP	000009EB-R	4174	(11)	2728	(6)		
ACT_SHOW	00000416-R	3022	(7)	2632	(6)		
ACT_SHOWLOAD	00000918-R	4076	(10)	2723	(6)		
ACT_SHOWMEM	000009E2-R	4170	(11)	2727	(6)		
ACT_SHOWMM	0000097E-R	4122	(10)	2651	(6)		
ACT_SHOWSECTIONS	000009D9-R	4163	(11)	2726	(6)		
ACT_SHOWSEL	000008F0-R	4048	(10)	2720	(6)		
ACT_SHOWSTATUS	000009C7-R	4148	(10)	2725	(6)		
ACT_SHOWSUP	000009D0-R	4156	(11)	2724	(6)		
ACT_SHOW_CALLS	0000063D-R	3436	(7)	2638	(6)		
ACT_SHOW_CRD_TRACE	0000065E-R	3460	(7)	2637	(6)		
ACT_START	0000041B-R	3030	(7)	2639	(6)		
ACT_SUBTEST	00000554-R	3217	(7)	2648	(6)		
ACT_SUCCESS	0000027A-R	2784	(7)	2610	(6)		
ACT_SUMMARY	00000440-R	3051	(7)	2640	(6)		

```
ACT_TEST                        00000559-R  3224  (7)     2646  (6)
ACT_TRACE                       000006E9-R  3594  (10)    2677  (6)
ACT_VERIFY                      000006EF-R  3602  (10)    2678  (6)
ACT_WIDTH                       00000713-R  3629  (10)    2688  (6)
ACT_WORD                        000007FF-R  3838  (10)    2701  (6)
ACT_XDELTA                      00000876-R  3983  (10)    2713  (6)
ADAPTER                        =0000006F    2717  (6)     1289  (4)     1492  (5)     866   (4)     898   (4)
ADDRESS                        =00000037    2661  (6)     1554  (5)     1743  (5)     1752  (5)     2146  (6)
                                                         2161  (6)     2301  (6)     2316  (6)
ADVANCE                        =0000005C    2698  (6)     699   (4)     745   (4)     792   (4)     821   (4)
ALL                            =00000039    2663  (6)     1549  (5)     1603  (5)     1638  (5)     1808  (5)
                                                         1942  (5)
APT_COM                         00000000-XR             2474  (6)     2475  (6)
APT_MSG                         00000000-XR             2470  (6)
ASCII                          =00000053    2689  (6)     2031  (5)
ATTACH                         =00000007    2613  (6)     460   (1)
BACKUP                         =0000005B    2697  (6)     1419  (4)     1431  (5)     1552  (5)     1557  (5)
                                                         1755  (5)     1835  (5)     1837  (5)     1888  (5)
                                                         2106  (6)     2117  (6)     2259  (6)     2270  (6)
                                                         520   (1)     522   (1)     524   (1)     550   (1)
                                                         552   (1)     579   (1)     581   (1)
BAD_LIST                        0000037D-R   771   (4)     1607  (5)     1646  (5)     1653  (5)     1663  (5)
                                                         1685  (5)     1812  (5)     1950  (5)     1960  (5)
                                                         1970  (5)     1974  (5)     1975  (5)     1993  (5)
                                                         2389  (6)
BAD_QUALIFIER                   0000039D-R   801   (4)     1153  (4)     1167  (4)     1294  (4)     1337  (4)
                                                         1436  (5)     1497  (5)     1579  (5)     1580  (5)
                                                         1581  (5)     1777  (5)     1778  (5)     1779  (5)
                                                         2082  (5)     2235  (6)     514   (1)     862   (4)
                                                          894  (4)
BASE                           =00000036    2660  (6)     1232  (4)     1741  (5)
BEGIN                           00000000-XR             #-2947  (7)
BEGINADAPTER                   =0000006E    2716  (6)     1287  (4)     1490  (5)     864   (4)     896   (4)
BELL                           =0000003B    2665  (6)     1643  (5)     1947  (5)
BINARY                         =0000003C    2666  (6)     1646  (5)     1950  (5)
BIT...                         =00000003    371   (1)      362  (1)      364   (1)     366   (1)     370   (1)
                                                          371  (1)
BREAK                          =00000038    2662  (6)     1239  (4)     1545  (5)     1750  (5)
BRIEF                          =0000006D    2715  (6)     1294  (4)     1436  (5)     1497  (5)
BYTE                           =00000060    2702  (6)     2038  (5)     2191  (6)     2354  (6)
B_SUCCESS                       0000044C-R   386   (1)     2785  (7)     2792  (7)
CLEAR                          =00000008    2614  (6)      478  (1)
CLEARENFORCE                   =00000033    2657  (6)     1595  (5)
CLEAR_CRD_DEBUG                =0000001E    2636  (6)     1582  (5)
CLEAR_CRD_TRACE                =0000001B    2633  (6)     1571  (5)
CLEAR_FLAGS_LOOP                000009AE-R  1636  (5)     1620  (5)     1626  (5)     1700  (5)
CLEAR_PARAMS                    000008D6-R  1534  (5)      479  (1)
CLI                             00000072-R  2485  (6)    #-2468  (6)
CLI$K_ALNUM                    =00000087    362   (1)     1059  (4)     1061  (4)     1068  (4)     1070  (4)
                                                         1081  (4)     1083  (4)     1090  (4)     1092  (4)
                                                         1289  (4)     1492  (5)     840   (4)     866   (4)
                                                          872  (4)      898  (4)     929   (4)     931   (4)
                                                          938  (4)      940  (4)     953   (4)     962   (4)
CLI$K_ALPHA                    =00000086    362   (1)     1046  (4)      916  (4)
CLI$K_BIF                      =00000083    362   (1)     1027  (4)     1101  (4)     2091  (5)     2245  (6)
                                                          672  (1)      851  (4)     869   (4)     876   (4)
                                                          878  (4)      982  (4)     983   (4)
```

```
CLI$K_BIFS                      =00000094     362    (1)
CLI$K_BR                        =00000082     362    (1)
```

```
1025  (4)     1029  (4)     1044  (4)     1057  (4)
1063  (4)     1065  (4)     1072  (4)     1074  (4)
1077  (4)     1085  (4)     1087  (4)     1094  (4)
1096  (4)     1103  (4)     1120  (4)     1130  (4)
1137  (4)     1148  (4)     1156  (4)     1172  (4)
1192  (4)     1233  (4)     1240  (4)     1249  (4)
1253  (4)     1267  (4)     1287  (4)     1292  (4)
1295  (4)     1298  (4)     1308  (4)     1315  (4)
1322  (4)     1332  (4)     1334  (4)     1336  (4)
1338  (4)     1345  (4)     1354  (4)     1363  (4)
1370  (4)     1377  (4)     1384  (4)     1391  (4)
1398  (4)     1419  (4)     1422  (4)     1431  (5)
1437  (5)     1439  (5)     1448  (5)     1455  (5)
1461  (5)     1490  (5)     1495  (5)     1500  (5)
1509  (5)     1512  (5)     1550  (5)     1552  (5)
1555  (5)     1557  (5)     1572  (5)     1583  (5)
1596  (5)     1605  (5)     1610  (5)     1620  (5)
1626  (5)     1639  (5)     1644  (5)     1647  (5)
1650  (5)     1655  (5)     1658  (5)     1661  (5)
1664  (5)     1667  (5)     1671  (5)     1674  (5)
1677  (5)     1680  (5)     1683  (5)     1686  (5)
1700  (5)     1702  (5)     1744  (5)     1753  (5)
1755  (5)     1770  (5)     1781  (5)     1788  (5)
1801  (5)     1810  (5)     1814  (5)     1823  (5)
1833  (5)     1835  (5)     1837  (5)     1848  (5)
1859  (5)     1862  (5)     1871  (5)     1872  (5)
1886  (5)     1888  (5)     1894  (5)     1897  (5)
1902  (5)     1907  (5)     1912  (5)     1917  (5)
1926  (5)     1943  (5)     1948  (5)     1951  (5)
1954  (5)     1957  (5)     1962  (5)     1965  (5)
1968  (5)     1971  (5)     1976  (5)     1979  (5)
1982  (5)     1985  (5)     1988  (5)     1991  (5)
1994  (5)     2003  (5)     2005  (5)     2032  (5)
2039  (5)     2046  (5)     2053  (5)     2060  (5)
2069  (5)     2076  (5)     2083  (5)     2104  (6)
2106  (6)     2115  (6)     2117  (6)     2125  (6)
2132  (6)     2139  (6)     2147  (6)     2155  (6)
2162  (6)     2168  (6)     2192  (6)     2199  (6)
2206  (6)     2213  (6)     2222  (6)     2229  (6)
2236  (6)     2257  (6)     2259  (6)     2268  (6)
2270  (6)     2279  (6)     2286  (6)     2293  (6)
2302  (6)     2310  (6)     2324  (6)     2330  (6)
2355  (6)     2362  (6)     2369  (6)     2376  (6)
2383  (6)     2390  (6)     2398  (6)     2404  (6)
454   (1)     479   (1)     486   (1)     516   (1)
520   (1)     522   (1)     524   (1)     531   (1)
539   (1)     540   (1)     541   (1)     548   (1)
550   (1)     552   (1)     559   (1)     577   (1)
579   (1)     581   (1)     588   (1)     606   (1)
614   (1)     623   (1)     630   (1)     640   (1)
646   (1)     652   (1)     659   (1)     666   (1)
674   (1)     683   (1)     699   (4)     702   (4)
731   (4)     745   (4)     783   (4)     785   (4)
792   (4)     812   (4)     814   (4)     821   (4)
838   (4)     842   (4)     847   (4)     849   (4)
852   (4)     864   (4)     871   (4)     874   (4)
```

				877 (4) 879 (4) 896 (4) 902 (4)
				914 (4) 927 (4) 933 (4) 935 (4)
				942 (4) 944 (4) 947 (4) 955 (4)
				957 (4) 964 (4) 966 (4) 985 (4)
CLI$K_BUFSIZ	=00000100	365	(1)	#-2497 (6) 365 (1)
CLI$K_CALL	=00000092	362	(1)	
CLI$K_COMMA	=0000008E	362	(1)	1054 (4) 1608 (5) 1699 (5) 1813 (5)
				2002 (5) 2397 (6) 782 (4) 845 (4)
				924 (4)
CLI$K_DEC	=0000008A	362	(1)	1099 (4) 1129 (4) 1155 (4) 1169 (4)
				1171 (4) 1607 (5) 1812 (5) 1847 (5)
				1870 (5) 1893 (5) 1901 (5) 1906 (5)
				1911 (5) 1916 (5) 1925 (5) 2068 (5)
				2146 (6) 2221 (6) 2301 (6) 622 (1)
				969 (4)
CLI$K_EOL	=00000091	362	(1)	1248 (4) 1252 (4) 1353 (4) 1571 (5)
				1582 (5) 1595 (5) 1769 (5) 1780 (5)
				1800 (5) 493 (1) 596 (1) 692 (1)
				735 (4) 787 (4) 816 (4)
CLI$K_ERROR	=00000080	362	(1)	705 (4) 736 (4) 752 (4) 764 (4)
				788 (4) 794 (4) 817 (4) 823 (4)
CLI$K_EXIT	=00000081	362	(1)	461 (1) 597 (1) 599 (1) 603 (1)
CLI$K_FILE	=00000095	362	(1)	951 (4) 960 (4)
CLI$K_HEX	=00000089	362	(1)	
CLI$K_KEYWORD	=0000008C	362	(1)	1024 (4) 1025 (4) 1027 (4) 1029 (4)
				1042 (4) 1044 (4) 1046 (4) 1047 (4)
				1049 (4) 1051 (4) 1052 (4) 1054 (4)
				1055 (4) 1057 (4) 1059 (4) 1061 (4)
				1062 (4) 1063 (4) 1064 (4) 1065 (4)
				1068 (4) 1068 (4) 1070 (4) 1072 (4)
				1072 (4) 1073 (4) 1074 (4) 1076 (4)
				1077 (4) 1079 (4) 1081 (4) 1083 (4)
				1084 (4) 1085 (4) 1086 (4) 1087 (4)
				1089 (4) 1090 (4) 1092 (4) 1093 (4)
				1094 (4) 1095 (4) 1096 (4) 1098 (4)
				1099 (4) 1101 (4) 1103 (4) 1119 (4)
				1120 (4) 1127 (4) 1128 (4) 1129 (4)
				1130 (4) 1136 (4) 1137 (4) 1143 (4)
				1145 (4) 1146 (4) 1147 (4) 1148 (4)
				1153 (4) 1154 (4) 1155 (4) 1156 (4)
				1167 (4) 1168 (4) 1169 (4) 1170 (4)
				1171 (4) 1172 (4) 1192 (4) 1224 (4)
				1226 (4) 1232 (4) 1233 (4) 1239 (4)
				1240 (4) 1246 (4) 1247 (4) 1248 (4)
				1249 (4) 1251 (4) 1252 (4) 1253 (4)
				1259 (4) 1260 (4) 1266 (4) 1267 (4)
				1281 (4) 1283 (4) 1285 (4) 1286 (4)
				1287 (4) 1289 (4) 1290 (4) 1292 (4)
				1294 (4) 1295 (4) 1297 (4) 1298 (4)
				1305 (4) 1306 (4) 1307 (4) 1308 (4)
				1314 (4) 1315 (4) 1321 (4) 1322 (4)
				1328 (4) 1329 (4) 1330 (4) 1331 (4)
				1332 (4) 1333 (4) 1334 (4) 1335 (4)
				1336 (4) 1337 (4) 1338 (4) 1344 (4)
				1345 (4) 1352 (4) 1353 (4) 1354 (4)
				1356 (4) 1357 (4) 1362 (4) 1363 (4)
				1369 (4) 1370 (4) 1376 (4) 1377 (4)

1383	(4)	1384	(4)	1390	(4)	1391	(4)
1397	(4)	1398	(4)	1401	(4)	1412	(4)
1413	(4)	1419	(4)	1421	(4)	1422	(4)
1431	(5)	1433	(5)	1434	(5)	1436	(5)
1437	(5)	1439	(5)	1447	(5)	1448	(5)
1454	(5)	1455	(5)	1460	(5)	1461	(5)
1482	(5)	1488	(5)	1489	(5)	1490	(5)
1492	(5)	1493	(5)	1495	(5)	1497	(5)
1499	(5)	1500	(5)	1506	(5)	1508	(5)
1509	(5)	1511	(5)	1512	(5)	1535	(5)
1543	(5)	1545	(5)	1546	(5)	1548	(5)
1549	(5)	1550	(5)	1552	(5)	1554	(5)
1555	(5)	1557	(5)	1567	(5)	1568	(5)
1569	(5)	1571	(5)	1572	(5)	1579	(5)
1580	(5)	1581	(5)	1582	(5)	1583	(5)
1593	(5)	1594	(5)	1595	(5)	1596	(5)
1597	(5)	1598	(5)	1600	(5)	1601	(5)
1603	(5)	1605	(5)	1607	(5)	1608	(5)
1610	(5)	1618	(5)	1619	(5)	1620	(5)
1626	(5)	1638	(5)	1639	(5)	1641	(5)
1643	(5)	1644	(5)	1646	(5)	1647	(5)
1649	(5)	1650	(5)	1652	(5)	1653	(5)
1654	(5)	1655	(5)	1657	(5)	1658	(5)
1660	(5)	1661	(5)	1663	(5)	1664	(5)
1666	(5)	1667	(5)	1670	(5)	1671	(5)
1673	(5)	1674	(5)	1676	(5)	1677	(5)
1679	(5)	1680	(5)	1682	(5)	1683	(5)
1685	(5)	1686	(5)	1699	(5)	1700	(5)
1702	(5)	1733	(5)	1735	(5)	1741	(5)
1742	(5)	1743	(5)	1744	(5)	1750	(5)
1751	(5)	1752	(5)	1753	(5)	1755	(5)
1765	(5)	1766	(5)	1767	(5)	1769	(5)
1770	(5)	1777	(5)	1778	(5)	1779	(5)
1780	(5)	1781	(5)	1787	(5)	1788	(5)
1798	(5)	1799	(5)	1800	(5)	1801	(5)
1803	(5)	1804	(5)	1805	(5)	1806	(5)
1808	(5)	1810	(5)	1812	(5)	1813	(5)
1814	(5)	1820	(5)	1821	(5)	1823	(5)
1829	(5)	1830	(5)	1831	(5)	1832	(5)
1833	(5)	1835	(5)	1837	(5)	1843	(5)
1845	(5)	1846	(5)	1847	(5)	1848	(5)
1855	(5)	1856	(5)	1857	(5)	1858	(5)
1859	(5)	1861	(5)	1862	(5)	1868	(5)
1869	(5)	1870	(5)	1871	(5)	1872	(5)
1884	(5)	1885	(5)	1886	(5)	1888	(5)
1891	(5)	1892	(5)	1893	(5)	1894	(5)
1896	(5)	1897	(5)	1899	(5)	1900	(5)
1901	(5)	1902	(5)	1904	(5)	1905	(5)
1906	(5)	1907	(5)	1909	(5)	1910	(5)
1911	(5)	1912	(5)	1914	(5)	1915	(5)
1916	(5)	1917	(5)	1923	(5)	1924	(5)
1925	(5)	1926	(5)	1942	(5)	1943	(5)
1945	(5)	1947	(5)	1948	(5)	1950	(5)
1951	(5)	1953	(5)	1954	(5)	1956	(5)
1957	(5)	1959	(5)	1960	(5)	1961	(5)
1962	(5)	1964	(5)	1965	(5)	1967	(5)
1968	(5)	1970	(5)	1971	(5)	1973	(5)

1974 (5) 1975 (5) 1976 (5) 1978 (5)
1979 (5) 1981 (5) 1982 (5) 1984 (5)
1985 (5) 1987 (5) 1988 (5) 1990 (5)
1991 (5) 1993 (5) 1994 (5) 2002 (5)
2003 (5) 2005 (5) 2025 (5) 2031 (5)
2032 (5) 2038 (5) 2039 (5) 2045 (5)
2046 (5) 2052 (5) 2053 (5) 2059 (5)
2060 (5) 2066 (5) 2067 (5) 2068 (5)
2069 (5) 2075 (5) 2076 (5) 2082 (5)
2083 (5) 2091 (5) 2094 (5) 2102 (6)
2103 (6) 2104 (6) 2106 (6) 2113 (6)
2114 (6) 2115 (6) 2117 (6) 2123 (6)
2124 (6) 2125 (6) 2131 (6) 2132 (6)
2138 (6) 2139 (6) 2145 (6) 2146 (6)
2147 (6) 2153 (6) 2154 (6) 2155 (6)
2161 (6) 2162 (6) 2168 (6) 2185 (6)
2191 (6) 2192 (6) 2198 (6) 2199 (6)
2205 (6) 2206 (6) 2212 (6) 2213 (6)
2219 (6) 2220 (6) 2221 (6) 2222 (6)
2228 (6) 2229 (6) 2235 (6) 2236 (6)
2245 (6) 2248 (6) 2255 (6) 2256 (6)
2257 (6) 2259 (6) 2266 (6) 2267 (6)
2268 (6) 2270 (6) 2277 (6) 2278 (6)
2279 (6) 2285 (6) 2286 (6) 2292 (6)
2293 (6) 2300 (6) 2301 (6) 2302 (6)
2308 (6) 2309 (6) 2310 (6) 2316 (6)
2322 (6) 2323 (6) 2324 (6) 2330 (6)
2348 (6) 2354 (6) 2355 (6) 2361 (6)
2362 (6) 2368 (6) 2369 (6) 2375 (6)
2376 (6) 2382 (6) 2383 (6) 2389 (6)
2390 (6) 2397 (6) 2398 (6) 2404 (6)
445 (1) 447 (1) 453 (1) 454 (1)
460 (1) 461 (1) 472 (1) 478 (1)
479 (1) 485 (1) 486 (1) 492 (1)
493 (1) 506 (1) 512 (1) 513 (1)
514 (1) 515 (1) 516 (1) 518 (1)
519 (1) 520 (1) 521 (1) 522 (1)
523 (1) 524 (1) 530 (1) 531 (1)
537 (1) 538 (1) 539 (1) 540 (1)
541 (1) 547 (1) 548 (1) 550 (1)
552 (1) 558 (1) 559 (1) 569 (1)
570 (1) 571 (1) 577 (1) 579 (1)
581 (1) 587 (1) 588 (1) 595 (1)
596 (1) 597 (1) 598 (1) 599 (1)
605 (1) 606 (1) 612 (1) 613 (1)
614 (1) 620 (1) 621 (1) 622 (1)
623 (1) 629 (1) 630 (1) 537 (1)
638 (1) 639 (1) 640 (1) 645 (1)
646 (1) 651 (1) 652 (1) 658 (1)
659 (1) 665 (1) 666 (1) 672 (1)
673 (1) 674 (1) 680 (1) 681 (1)
683 (1) 692 (1) 693 (1) 696 (2)
699 (4) 702 (4) 705 (4) 731 (4)
735 (4) 736 (4) 743 (4) 745 (4)
752 (4) 764 (4) 782 (4) 783 (4)
785 (4) 787 (4) 788 (4) 790 (4)
792 (4) 794 (4) 811 (4) 812 (4)

```
                                                        814     (4)     816     (4)     817     (4)     819     (4)
                                                        821     (4)     823     (4)     836     (4)     838     (4)
                                                        840     (4)     841     (4)     842     (4)     843     (4)
                                                        845     (4)     847     (4)     849     (4)     851     (4)
                                                        852     (4)     860     (4)     862     (4)     863     (4)
                                                        864     (4)     866     (4)     867     (4)     869     (4)
                                                        870     (4)     871     (4)     872     (4)     873     (4)
                                                        874     (4)     875     (4)     876     (4)     877     (4)
                                                        878     (4)     879     (4)     892     (4)     894     (4)
                                                        895     (4)     896     (4)     898     (4)     899     (4)
                                                        901     (4)     902     (4)     914     (4)     916     (4)
                                                        917     (4)     919     (4)     921     (4)     922     (4)
                                                        924     (4)     925     (4)     927     (4)     929     (4)
                                                        931     (4)     932     (4)     933     (4)     934     (4)
                                                        935     (4)     937     (4)     938     (4)     940     (4)
                                                        941     (4)     942     (4)     943     (4)     944     (4)
                                                        946     (4)     947     (4)     949     (4)     951     (4)
                                                        953     (4)     954     (4)     955     (4)     956     (4)
                                                        957     (4)     959     (4)     960     (4)     962     (4)
                                                        963     (4)     964     (4)     965     (4)     966     (4)
                                                        968     (4)     969     (4)     982     (4)     983     (4)
                                                        985     (4)
CLI$K_NUM               =00000085       362     (1)     1554    (5)     1743    (5)     1752    (5)     2138    (6)
                                                        2161    (6)     2292    (6)     2316    (6)     2323    (6)
CLI$K_OCT               =00000088       362     (1)     1052    (4)     1055    (4)     922     (4)     925     (4)
CLI$K_RETURN            =00000C93       362     (1)
CLI$K_SIZE               00000444       365     (1)     2453    (6)     2454    (6)     2457    (6)     2460    (6)
                                                      #-2463    (6)     381     (1)
CLI$K_SLASH             =00000090       362     (1)     1024    (4)     1283    (4)     1330    (4)     1434    (5)
                                                        1482    (5)     1499    (5)     1568    (5)     1580    (5)
                                                        1766    (5)     1778    (5)     2025    (5)     2185    (6)
                                                         513    (1)      811    (4)      860    (4)      892    (4)
CLI$K_SPACE             =00000084       362     (1)     1042    (4)     1224    (4)     1297    (4)     1306    (4)
                                                        1506    (5)     1535    (5)     1546    (5)     1598    (5)
                                                        1601    (5)     1619    (5)     1733    (5)     1742    (5)
                                                        1751    (5)     1804    (5)     1806    (5)     1821    (5)
                                                        1832    (5)     1846    (5)     1856    (5)     1869    (5)
                                                        1892    (5)     1900    (5)     1905    (5)     1910    (5)
                                                        1915    (5)     1924    (5)     2094    (5)     2248    (6)
                                                        2322    (6)     2348    (6)      445    (1)      515    (1)
                                                         598    (1)      613    (1)      621    (1)      681    (1)
                                                         696    (2)      743    (4)      790    (4)      819    (4)
                                                         836    (4)      870    (4)      901    (4)
CLI$K_STRING            =00000088       362     (1)     1024    (4)     1025    (4)     1027    (4)     1029    (4)
                                                        1042    (4)     1044    (4)     1046    (4)     1047    (4)
                                                        1049    (4)     1051    (4)     1052    (4)     1054    (4)
                                                        1055    (4)     1057    (4)     1059    (4)     1061    (4)
                                                        1062    (4)     1063    (4)     1064    (4)     1065    (4)
                                                        1067    (4)     1068    (4)     1070    (4)     1071    (4)
                                                        1072    (4)     1073    (4)     1074    (4)     1076    (4)
                                                        1077    (4)     1079    (4)     1081    (4)     1083    (4)
                                                        1084    (4)     1085    (4)     1086    (4)     1087    (4)
                                                        1089    (4)     1090    (4)     1092    (4)     1093    (4)
                                                        1094    (4)     1095    (4)     1096    (4)     1098    (4)
                                                        1099    (4)     1101    (4)     1103    (4)     1119    (4)
                                                        1120    (4)     1127    (4)     1128    (4)     1129    (4)
                                                        1130    (4)     1136    (4)     1137    (4)     1143    (4)
```

1145 (4) 1146 (4) 1147 (4) 1148 (4)
1153 (4) 1154 (4) 1155 (4) 1156 (4)
1167 (4) 1168 (4) 1169 (4) 1170 (4)
1171 (4) 1172 (4) 1192 (4) 1224 (4)
1226 (4) 1232 (4) 1233 (4) 1239 (4)
1240 (4) 1246 (4) 1247 (4) 1248 (4)
1249 (4) 1251 (4) 1252 (4) 1253 (4)
1259 (4) 1260 (4) 1266 (4) 1267 (4)
1281 (4) 1283 (4) 1285 (4) 1286 (4)
1287 (4) 1289 (4) 1290 (4) 1292 (4)
1294 (4) 1295 (4) 1297 (4) 1298 (4)
1305 (4) 1306 (4) 1307 (4) 1308 (4)
1314 (4) 1315 (4) 1321 (4) 1322 (4)
1328 (4) 1329 (4) 1330 (4) 1331 (4)
1332 (4) 1333 (4) 1334 (4) 1335 (4)
1336 (4) 1337 (4) 1338 (4) 1344 (4)
1345 (4) 1352 (4) 1353 (4) 1354 (4)
1356 (4) 1357 (4) 1362 (4) 1363 (4)
1369 (4) 1370 (4) 1376 (4) 1377 (4)
1383 (4) 1384 (4) 1390 (4) 1391 (4)
1397 (4) 1398 (4) 1401 (4) 1412 (4)
1413 (4) 1419 (4) 1421 (4) 1422 (4)
1431 (5) 1433 (5) 1434 (5) 1436 (5)
1437 (5) 1439 (5) 1447 (5) 1448 (5)
1454 (5) 1455 (5) 1460 (5) 1461 (5)
1482 (5) 1488 (5) 1489 (5) 1490 (5)
1492 (5) 1493 (5) 1495 (5) 1497 (5)
1499 (5) 1500 (5) 1506 (5) 1508 (5)
1509 (5) 1511 (5) 1512 (5) 1535 (5)
1543 (5) 1545 (5) 1546 (5) 1548 (5)
1549 (5) 1550 (5) 1552 (5) 1554 (5)
1555 (5) 1557 (5) 1567 (5) 1568 (5)
1569 (5) 1571 (5) 1572 (5) 1579 (5)
1580 (5) 1581 (5) 1582 (5) 1583 (5)
1593 (5) 1594 (5) 1595 (5) 1596 (5)
1597 (5) 1598 (5) 1600 (5) 1601 (5)
1603 (5) 1605 (5) 1607 (5) 1608 (5)
1610 (5) 1618 (5) 1619 (5) 1620 (5)
1626 (5) 1638 (5) 1639 (5) 1641 (5)
1643 (5) 1644 (5) 1646 (5) 1647 (5)
1649 (5) 1650 (5) 1652 (5) 1653 (5)
1654 (5) 1655 (5) 1657 (5) 1658 (5)
1660 (5) 1661 (5) 1663 (5) 1664 (5)
1666 (5) 1667 (5) 1670 (5) 1671 (5)
1673 (5) 1674 (5) 1676 (5) 1677 (5)
1679 (5) 1680 (5) 1682 (5) 1683 (5)
1685 (5) 1686 (5) 1699 (5) 1700 (5)
1702 (5) 1733 (5) 1735 (5) 1741 (5)
1742 (5) 1743 (5) 1744 (5) 1750 (5)
1751 (5) 1752 (5) 1753 (5) 1755 (5)
1765 (5) 1766 (5) 1767 (5) 1769 (5)
1770 (5) 1777 (5) 1778 (5) 1779 (5)
1780 (5) 1781 (5) 1787 (5) 1788 (5)
1798 (5) 1799 (5) 1800 (5) 1801 (5)
1803 (5) 1804 (5) 1805 (5) 1806 (5)
1808 (5) 1810 (5) 1812 (5) 1813 (5)
1814 (5) 1820 (5) 1821 (5) 1823 (5)

1829	(5)	1830	(5)	1831	(5)	1832	(5)
1833	(5)	1835	(5)	1837	(5)	1843	(5)
1845	(5)	1846	(5)	1847	(5)	1848	(5)
1855	(5)	1856	(5)	1857	(5)	1858	(5)
1859	(5)	1861	(5)	1862	(5)	1868	(5)
1869	(5)	1870	(5)	1871	(5)	1872	(5)
1884	(5)	1885	(5)	1886	(5)	1888	(5)
1891	(5)	1892	(5)	1893	(5)	1894	(5)
1896	(5)	1897	(5)	1899	(5)	1900	(5)
1901	(5)	1902	(5)	1904	(5)	1905	(5)
1906	(5)	1907	(5)	1909	(5)	1910	(5)
1911	(5)	1912	(5)	1914	(5)	1915	(5)
1916	(5)	1917	(5)	1923	(5)	1924	(5)
1925	(5)	1926	(5)	1942	(5)	1943	(5)
1945	(5)	1947	(5)	1948	(5)	1950	(5)
1951	(5)	1953	(5)	1954	(5)	1956	(5)
1957	(5)	1959	(5)	1960	(5)	1961	(5)
1962	(5)	1964	(5)	1965	(5)	1967	(5)
1968	(5)	1970	(5)	1971	(5)	1973	(5)
1974	(5)	1975	(5)	1976	(5)	1978	(5)
1979	(5)	1981	(5)	1982	(5)	1984	(5)
1985	(5)	1987	(5)	1988	(5)	1990	(5)
1991	(5)	1993	(5)	1994	(5)	2002	(5)
2003	(5)	2005	(5)	2025	(5)	2031	(5)
2032	(5)	2038	(5)	2039	(5)	2045	(5)
2046	(5)	2052	(5)	2053	(5)	2059	(5)
2060	(5)	2066	(5)	2067	(5)	2068	(5)
2069	(5)	2075	(5)	2076	(5)	2082	(5)
2083	(5)	2091	(5)	2094	(5)	2102	(6)
2103	(6)	2104	(6)	2106	(6)	2113	(6)
2114	(6)	2115	(6)	2117	(6)	2123	(6)
2124	(6)	2125	(6)	2131	(6)	2132	(6)
2138	(6)	2139	(6)	2145	(6)	2146	(6)
2147	(6)	2153	(6)	2154	(6)	2155	(6)
2161	(6)	2162	(6)	2168	(6)	2185	(6)
2191	(6)	2192	(6)	2198	(6)	2199	(6)
2205	(6)	2206	(6)	2212	(6)	2213	(6)
2219	(6)	2220	(6)	2221	(6)	2222	(6)
2228	(6)	2229	(6)	2235	(6)	2236	(6)
2245	(6)	2248	(6)	2255	(6)	2256	(6)
2257	(6)	2259	(6)	2266	(6)	2267	(6)
2268	(6)	2270	(6)	2277	(6)	2278	(6)
2279	(6)	2285	(6)	2286	(6)	2292	(6)
2293	(6)	2300	(6)	2301	(6)	2302	(6)
2308	(6)	2309	(6)	2310	(6)	2316	(6)
2322	(6)	2323	(6)	2324	(6)	2330	(6)
2348	(6)	2354	(6)	2355	(6)	2361	(6)
2362	(6)	2368	(6)	2369	(6)	2375	(6)
2376	(6)	2382	(6)	2383	(6)	2389	(6)
2390	(6)	2397	(6)	2398	(6)	2404	(6)
445	(1)	447	(1)	453	(1)	454	(1)
460	(1)	461	(1)	472	(1)	478	(1)
479	(1)	485	(1)	486	(1)	492	(1)
493	(1)	506	(1)	512	(1)	513	(1)
514	(1)	515	(1)	516	(1)	518	(1)
519	(1)	520	(1)	521	(1)	522	(1)
523	(1)	524	(1)	530	(1)	531	(1)

				537	(1)	538	(1)	539	(1)	540	(1)		
				541	(1)	547	(1)	548	(1)	550	(1)		
				552	(1)	558	(1)	559	(1)	569	(1)		
				570	(1)	571	(1)	577	(1)	579	(1)		
				581	(1)	587	(1)	598	(1)	595	(1)		
				596	(1)	597	(1)	598	(1)	599	(1)		
				605	(1)	606	(1)	612	(1)	613	(1)		
				614	(1)	620	(1)	621	(1)	622	(1)		
				623	(1)	629	(1)	630	(1)	637	(1)		
				638	(1)	639	(1)	640	(1)	645	(1)		
				646	(1)	651	(1)	652	(1)	658	(1)		
				659	(1)	665	(1)	666	(1)	672	(1)		
				673	(1)	674	(1)	680	(1)	681	(1)		
				683	(1)	692	(1)	693	(1)	696	(2)		
				699	(4)	702	(4)	705	(4)	731	(4)		
				735	(4)	736	(4)	743	(4)	745	(4)		
				752	(4)	764	(4)	782	(4)	783	(4)		
				785	(4)	787	(4)	788	(4)	790	(4)		
				792	(4)	794	(4)	811	(4)	812	(4)		
				814	(4)	816	(4)	817	(4)	819	(4)		
				821	(4)	823	(4)	836	(4)	838	(4)		
				840	(4)	841	(4)	842	(4)	843	(4)		
				845	(4)	847	(4)	849	(4)	851	(4)		
				852	(4)	860	(4)	862	(4)	863	(4)		
				864	(4)	866	(4)	867	(4)	869	(4)		
				870	(4)	871	(4)	872	(4)	873	(4)		
				874	(4)	875	(4)	876	(4)	877	(4)		
				878	(4)	879	(4)	892	(4)	894	(4)		
				895	(4)	896	(4)	898	(4)	899	(4)		
				901	(4)	902	(4)	914	(4)	916	(4)		
				917	(4)	919	(4)	921	(4)	922	(4)		
				924	(4)	925	(4)	927	(4)	929	(4)		
				931	(4)	932	(4)	933	(4)	934	(4)		
				935	(4)	937	(4)	938	(4)	940	(4)		
				941	(4)	942	(4)	943	(4)	944	(4)		
				946	(4)	947	(4)	949	(4)	951	(4)		
				953	(4)	954	(4)	955	(4)	956	(4)		
				957	(4)	959	(4)	960	(4)	962	(4)		
				963	(4)	964	(4)	965	(4)	966	(4)		
				968	(4)	969	(4)	982	(4)	983	(4)		
				985	(4)								
CLI$K_SYMB	=0000008D	362	(1)	1047	(4)	1147	(4)	917	(4)				
CLI$K_VALU	=0000008F	362	(1)	1128	(4)	1146	(4)	1154	(4)	1168	(4)		
				1170	(4)	1286	(4)	1489	(5)	2067	(5)		
				2220	(6)	863	(4)	895	(4)				
CLI$L_ADDRESS	00000018	365	(1)	#-2975	(7)	#-3365	(7)	#-3765	(10)	#-3768	(10)		
				#-3771	(10)	#-3778	(10)	#-3782	(10)				
CLI$L_COMMAND	00000004	365	(1)	#-2474	(6)	#-2569	(6)	#-2818	(7)	#-2826	(7)		
				#-2834	(7)	#-2844	(7)	#-2862	(7)	#-2870	(7)		
				#-2885	(7)	#-2894	(7)	#-2903	(7)	#-2916	(7)		
				#-2926	(7)	#-2954	(7)	#-2972	(7)	#-2984	(7)		
				#-2994	(7)	#-3006	(7)	#-3014	(7)	#-3034	(7)		
				#-3045	(7)	#-3053	(7)	#-3308	(7)	#-3321	(7)		
				#-3326	(7)	#-3329	(7)	#-3343	(7)	#-3355	(7)		
				#-3358	(7)	#-3379	(7)	#-3384	(7)	#-3387	(7)		
				#-3397	(7)	#-3403	(7)	#-3406	(7)	#-3419	(7)		
				#-3423	(7)	#-3438	(7)	#-3473	(7)	#-3618	(10)		

```
                                                #-3621  (10)    #-3623  (10)    #-3635  (10)    #-3638  (10)
                                                #-3640  (10)    #-3681  (10)    #-3685  (10)    #-3688  (10)
                                                #-3706  (10)    #-3711  (10)    #-3714  (10)    #-3985  (10)
                                                #-3998  (10)    #-4001  (10)    #-4011  (10)    #-4013  (10)
                                                #-4016  (10)    #-4018  (10)    #-4051  (10)    #-4054  (10)
                                                #-4061  (10)    #-4070  (10)    #-4077  (10)    #-4086  (10)
                                                #-4105  (10)    #-4123  (10)    #-4149  (10)    #-4157  (11)
                                                #-4164  (11)    #-4171  (11)    #-4194  (11)
CLI$L_DATA              0000001C    365 (1)     #-2982  (7)     #-3396  (7)     #-3439  (7)       3494  (8)
                                                  3504  (10)      3512  (10)      3520  (10)      3528  (10)
                                                  3536  (10)      3544  (10)      3554  (10)      3562  (10)
                                                  3570  (10)      3578  (10)      3588  (10)      3596  (10)
                                                  3604  (10)      3727  (10)    #-3822  (10)
CLI$L_FLAGS             00000000    365 (1)       2541  (6)       2758  (7)       2766  (7)       2800  (7)
                                                  2808  (7)       2836  (7)       2887  (7)       2918  (7)
                                                #-2974  (7)       2992  (7)       3016  (7)       3024  (7)
                                                  3032  (7)       3169  (7)       3177  (7)       3202  (7)
                                                  3317  (7)       3319  (7)       3324  (7)       3337  (7)
                                                  3339  (7)       3341  (7)       3353  (7)       3367  (7)
                                                  3375  (7)       3377  (7)       3382  (7)       3401  (7)
                                                  3414  (7)       3417  (7)       3427  (7)       3614  (10)
                                                  3616  (10)      3631  (10)      3633  (10)      3651  (10)
                                                  3656  (10)      3661  (10)      3666  (10)      3671  (10)
                                                  3676  (10)      3678  (10)      3695  (10)      3701  (10)
                                                  3703  (10)      3751  (10)      3761  (10)      3785  (10)
                                                  3788  (10)      3795  (10)      3832  (10)      3840  (10)
                                                  3848  (10)      3856  (10)      3864  (10)      3872  (10)
                                                  3889  (10)      3905  (10)      3947  (10)      3951  (10)
                                                  3996  (10)      4008  (10)      4040  (10)      4049  (10)
CLI$L_LAST              00000024    365 (1)     #-3232  (7)
CLI$L_NEXT              00000030    365 (1)     #-3743  (10)
CLI$L_PASS              0000002C    365 (1)     #-2995  (7)     #-2997  (7)     #-3035  (7)     #-3037  (7)
                                                #-3194  (7)
CLI$L_SUBT              00000028    365 (1)     #-3218  (7)
CLI$L_TEST              00000020    365 (1)     #-3225  (7)
CLI$Q_BUFQWD            00000034    365 (1)       2498  (6)     #-2518  (6)       2527  (6)     #-2774  (7)
CLI$Q_FILE              00000008    365 (1)     #-2816  (7)     #-2817  (7)     #-2874  (7)     #-2876  (7)
                                                #-3152  (7)     #-3153  (7)     #-3160  (7)     #-3161  (7)
                                                #-3890  (10)    #-3918  (10)
CLI$Q_SECTION           00000010    365 (1)     #-3210  (7)     #-3211  (7)
CLI$Q_TIME              0000043C    365 (1)
CLI$T_BUFFER            0000003C    365 (1)       2499  (6)       2517  (6)
CLI$V_ADAPTER           =00000018   365 (1)     #-4039  (10)
CLI$V_ADR               =00C00008   365 (1)     #-3366  (7)     #-3787  (10)    #-3950  (10)
CLI$V_ASCII             =00000013   365 (1)     #-3750  (10)
CLI$V_BREAK             =0000000A   365 (1)     #-3374  (7)
CLI$V_BRIEF             =00000018   365 (1)     #-3996  (10)    #-4008  (10)    #-4049  (10)
CLI$V_BYTE              =0000000D   365 (1)     #-3847  (10)
CLI$V_CLEAR             =00000002   365 (1)     #-2835  (7)     #-3323  (7)     #-3381  (7)
CLI$V_DEC               =00000010   365 (1)     #-3863  (10)
CLI$V_DEFAULT           =0000000C   365 (1)     #-3426  (7)     #-3694  (10)
CLI$V_DEPOSIT           =00000019   365 (1)     #-2886  (7)
CLI$V_EVENT             =00000008   365 (1)     #-3316  (7)     #-3336  (7)
CLI$V_EXAM              =00000005   365 (1)     #-2917  (7)
CLI$V_FLAGS             =00000009   365 (1)     #-3338  (7)     #-3413  (7)
CLI$V_HEX               =00000012   365 (1)     #-3855  (10)
CLI$V_KERNEL            =00000017   365 (1)
```

```
CLI$V_LOAD                      =00000006       365     (1)     #-2973    (7)
CLI$V_LONG                      =0000000F       365     (1)     #-3831    (10)
CLI$V_NOTNUF                    =00000001       365     (1)     #-2540    (6)     #-2756    (6)     #-2765    (7)     #-2799    (7)
                                                                #-2807    (7)
CLI$V_OCT                       =00000011       365     (1)     #-3871    (10)
CLI$V_PREG                      =0000001A       365     (1)     #-3795    (10)
CLI$V_QA                        =00000007       365     (1)     #-3201    (7)
CLI$V_QACKLOOPLOOPS             =0000001C       365     (1)     #-3655    (10)
CLI$V_QAERRORPRINTS             =0000001B       365     (1)     #-3650    (10)
CLI$V_QAMULTIPLEPASS            =0000001F       365     (1)     #-3670    (10)
CLI$V_QASUBTESTLOOPS            =0000001E       365     (1)     #-3665    (10)
CLI$V_QATESTLOOPS               =0000001D       365     (1)     #-3660    (10)
CLI$V_REG                       =00000014       365     (1)     #-3760    (10)    #-3784    (10)    #-3946    (10)
CLI$V_REQUIRED                  =00000000       365     (1)     #-3168    (7)     #-3176    (7)     #-3904    (10)
CLI$V_RUN                       =00000015       365     (1)     #-2991    (7)     #-3031    (7)     #-3888    (10)
CLI$V_SET                       =00000003       365     (1)     #-3015    (7)     #-3318    (7)     #-3376    (7)     #-3400    (7)
                                                                #-3615    (10)    #-3632    (10)    #-3677    (10)    #-3702    (10)
CLI$V_SHOW                      =00000004       365     (1)     #-3023    (7)     #-3340    (7)     #-3352    (7)     #-3416    (7)
                                                                #-3613    (10)    #-3630    (10)    #-3675    (10)    #-3700    (10)
CLI$V_VALSEC                    =00000016       365     (1)
CLI$V_WORD                      =0000000E       365     (1)     #-3839    (10)
CLI_ACTION                       0000013D-R     2602    (6)       2525    (6)
COMMAND_TREE                     00000180-R     444     (1)       2526    (6)
COMMON_CRD_TRACE_DEBUG           00000689-R     3472    (7)     #-3452    (7)     #-3458    (7)
COMMON_ENFORCE                   0000056A-R     3306    (7)     #-3303    (7)
CONTEXT                         =00000003       2609    (6)
CONTINUE                        =00000009       2615    (6)       485     (1)
CRD                             =00000023       2641    (6)       493     (1)
CRD$GL_CRD_COMMAND_DEBUG         00000000-XR                    #-3121    (7)
CRD$K_CRD_INITIALIZATION        =00000000       371     (1)     #-3095    (7)
CRD$K_DS_CONTROL_C              =00000001       371     (1)
CRD$K_DS_FLAGS                  =00000000       371     (1)
CRD$K_FAILURE                   =00000000       371     (1)
CRD$K_HOOK_POINT                =00000000       371     (1)     #-3096    (7)
CRD$K_LAST_ACCESS_CODE          =00000002       371     (1)
CRD$K_LAST_DS_VARIABLE          =00000002       371     (1)
CRD$K_LAST_FUNCTION             =00000002       371     (1)
CRD$K_LAST_HOOK_POINT           =00000001       371     (1)
CRD$K_READ                      =00000000       371     (1)
CRD$K_SUCCESS                   =00000001       371     (1)
CRD$K_TYPECODE                  =00000001       371     (1)
CRD$K_WRITE                     =00000001       371     (1)
DATA                            =0000005D       2699    (6)      1847    (5)      1870    (5)      1893    (5)      1901    (5)
                                                                 1906    (5)      1911    (5)      1916    (5)      1925    (5)
                                                                 2323    (6)       622    (1)
DBG_NAME                         0000030D-R     2878    (7)     #-2874    (7)      2875    (7)
DEATTACH                        =0000000A       2616    (6)       530     (1)
DEATTACH_QUAL                    000003E9-R     859     (4)       531     (1)      877     (4)
DEBUG                           =0000000B       2617    (6)       537     (1)
DEBUG$GB_FLAGS                   00000000-XR                    #-2843    (7)     #-2983    (7)
DEBUG_SWITCH                    =0000000C       2618    (6)      1119     (4)
DEC                             =00000062       2704    (6)      2045     (5)      2198    (6)      2361    (6)
DEFAULT                         =0000003A       2664    (6)      1266     (4)      1787    (5)      1953    (5)
DEFAULT_DBG                     =0000000D       2619    (6)       540     (1)
DEPOSIT                         =0000000E       2620    (6)       558     (1)
DEPOSIT_PARAMS                   00000E76-R     2184    (6)      2192     (6)      2199    (6)      2206    (6)      2213    (6)
                                                                 2222     (6)      2229    (6)      2236    (6)      2324    (6)
```

```
                                                        559     (1)
DESELECT                        =00000070       2718    (6)     547     (1)
DESELECT_QUALS                   00000435-R     891     (4)     548     (1)
DEVICE                          =0000006C       2714    (6)     1281    (4)     1508    (5)
DIRECTORY                       =0000000F       2621    (6)     512     (1)
DIR_WIDE                        =00000080       2734    (6)     514     (1)
DO_CMD                          =00000073       2721    (6)     845     (4)
DS$A_PRGBGN                      00000000-XR                    #-2975  (7)
DS$CLI                           00000000-R     2449    (6)
DS$GA_CRD_DS_INTERFACE           00000000-XR                    3098    (7)
DS$GA_DS_CTRL_C_FIRST            00000000-XR                    #-3075  (7)
DS$GA_DS_CTRL_C_SECOND           00000000-XR                    #-3076  (7)
DS$GB_INHIBIT_NAMING             00000000-XR                    #-3307  (7)
DS$GB_MM_ENB                     00000000-XR                    #-4096  (10)    #-4115  (10)    #-4133  (10)
DS$GL_CLIBASE                    00000000-R     380     (1)     2453    (6)
DS$GL_CRD_DEBUG                  00000000-XR                    #-3478  (7)     #-3483  (7)
DS$GL_FLAGS                      00000000-XR                    2469    (6)     2477    (6)     2494    (6)     3080    (7)
                                                                3083    (7)
DS$GT_PROMPT                     00000000-XR                    2480    (6)     2496    (6)
DS$K_ERROR                      =00000002       364     (1)
DS$K_NORMAL                     =00000001       364     (1)
DS$K_PRINTB                     =00000002       370     (1)
DS$K_PRINTF                     =00000001       370     (1)     #-2850  (7)     #-2959  (7)     #-4091  (10)    #-4110  (10)
                                                                #-4129  (10)    #-4138  (10)    #-4200  (11)
DS$K_PRINTI                     =00000000       370     (1)     #-2480  (6)     #-2549  (6)     #-2560  (6)     #-3109  (7)
                                                                #-3138  (7)     #-3470  (7)     #-3730  (10)
DS$K_PRINTX                     =00000003       370     (1)
DS$K_SEVERE                     =00000004       364     (1)
DS$K_SUBSYS                     =00000066       364     (1)     364     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014       370     (1)
DS$K_TYPE_ABORT_TEST            =00000013       370     (1)
DS$K_TYPE_COMMAND_ERR           =00000015       370     (1)     #-2550  (6)     #-2561  (6)     #-2960  (7)     #-3109  (7)
                                                                #-3138  (7)     #-3731  (10)    #-4092  (10)    #-4111  (10)
                                                                #-4129  (10)    #-4201  (11)
DS$K_TYPE_COMMAND_OUT           =00000016       370     (1)     #-2850  (7)     #-3470  (7)     #-4139  (10)
DS$K_TYPE_CRD_AUTOTEST          =0000001A       370     (1)
DS$K_TYPE_DS_PROMPT             =00000001       370     (1)     #-2480  (6)
DS$K_TYPE_DS_START              =0000001D       370     (1)
DS$K_TYPE_ERRDEV                =00000008       370     (1)
DS$K_TYPE_ERRHARD               =00000006       370     (1)
DS$K_TYPE_ERROR_BODY            =00000009       370     (1)
DS$K_TYPE_ERROR_END             =0000000A       370     (1)
DS$K_TYPE_ERRPREP               =0000001B       370     (1)
DS$K_TYPE_ERRSOFT               =00000007       370     (1)
DS$K_TYPE_ERRSUP                =00000004       370     (1)
DS$K_TYPE_ERRSYS                =00000005       370     (1)
DS$K_TYPE_ERR_HALT              =0000000D       370     (1)
DS$K_TYPE_EXCEPTION             =0000000C       370     (1)
DS$K_TYPE_EXCEPTION_HEAD        =00000006       370     (1)
DS$K_TYPE_FIRST_PASS            =00000011       370     (1)
DS$K_TYPE_GENERAL               =00000000       370     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003       370     (1)
DS$K_TYPE_NO_TESTS              =00000012       370     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C       370     (1)
DS$K_TYPE_PROGRAM_END           =00000010       370     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017       370     (1)
DS$K_TYPE_PROGRAM_START         =0000000F       370     (1)
```

```
DS$K_TYPE_QIO_INVADP            =00000024      370     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      370     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      370     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      370     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      370     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      370     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      370     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      370     (1)
DS$K_TYPE_START_ERR             =00000018      370     (1)
DS$K_TYPE_START_LIST            =00000025      370     (1)
DS$K_TYPE_SUMMARY               =0000000E      370     (1)
DS$K_TYPE_USER_PROMPT           =00000002      370     (1)
DS$K_WARNING                    =00000000      364     (1)
DS$L_USERCNTRLC                  00000000-XR                    #-3077   (7)
DS$M_ABRTFLG                    =00000040      366     (1)
DS$M_BADTIME                    =00100000      366     (1)
DS$M_BATCH                      =00400000      366     (1)
DS$M_BRKCLR                     =00001000      366     (1)
DS$M_BRKPT                      =00000800      366     (1)
DS$M_CHARFLG                    =00000100      366     (1)
DS$M_CMDFLG                     =00000080      366     (1)
DS$M_CTRLC                      =00000001      366     (1)
DS$M_CTRLO                      =00010000      366     (1)
DS$M_DEVFLG                     =00000200      366     (1)
DS$M_DISABLCC                   =01000000      366     (1)
DS$M_DONFLG                     =00002000      366     (1)
DS$M_ERRFLG                     =00000010      366     (1)
DS$M_EXCEPT                     =00080000      366     (1)
DS$M_EXETST                     =00040000      366     (1)
DS$M_HLTFLG                     =00000008      366     (1)
DS$M_LODFLG                     =00000002      366     (1)
DS$M_MEMMGT                     =00008000      366     (1)
DS$M_OUTPUT                     =00800000      366     (1)
DS$M_RUBFLG                     =00000020      366     (1)
DS$M_SCRIPT                     =00200000      366     (1)
DS$M_SETIMR                     =02000000      366     (1)
DS$M_STRFLG                     =00000004      366     (1)
DS$M_SUBT                       =00004000      366     (1)
DS$M_SYSFLG                     =00000400      366     (1)
DS$M_TIMRON                     =00020000      366     (1)
DS$V_ABRTFLG                    =00000006      366     (1)
DS$V_BADTIME                    =00000014      366     (1)
DS$V_BATCH                      =00000016      366     (1)
DS$V_BRKCLR                     =0000000C      366     (1)
DS$V_BRKPT                      =0000000B      366     (1)
DS$V_CHARFLG                    =00000008      366     (1)
DS$V_CMDFLG                     =00000007      366     (1)
DS$V_CTRLC                      =00000000      366     (1)    #-2469   (6)    #-2494   (6)    #-3079   (7)
DS$V_CTRLO                      =00000010      366     (1)    #-2477   (6)
DS$V_DEVFLG                     =00000009      366     (1)
DS$V_DISABLCC                   =00000018      366     (1)    #-3082   (7)
DS$V_DONFLG                     =0000000D      366     (1)
DS$V_ERRFLG                     =00000004      366     (1)
DS$V_EXCEPT                     =00000013      366     (1)
DS$V_EXETST                     =00000012      366     (1)
DS$V_HLTFLG                     =00000003      366     (1)
DS$V_LODFLG                     =00000001      366     (1)
```

```
DS$V_MEMMGT                     =0000000F       366     (1)
DS$V_OUTPUT                     =00000017       366     (1)
DS$V_RUBFLG                     =00000005       366     (1)
DS$V_SCRIPT                     =00000015       366     (1)
DS$V_SETIMR                     =00000019       366     (1)
DS$V_STRFLG                     =00000002       366     (1)
DS$V_SUBT                       =0000000E       366     (1)
DS$V_SYSFLG                     =0000000A       366     (1)
DS$V_TIMRON                     =00000011       366     (1)
DS$_ARITH                       =006600D0       364     (1)
DS$_ASBE                        =00660118       364     (1)
DS$_BADLINK                     =006600F0       364     (1)
DS$_BADTYPE                     =006600E8       364     (1)
DS$_BIIC                        =00660120       364     (1)
DS$_CHME                        =006600A8       364     (1)
DS$_CHMK                        =006600E0       364     (1)
DS$_DEVNAME                     =00660108       364     (1)
DS$_ERROR                       =00660002       364     (1)
DS$_FHWE                        =00660068       364     (1)
DS$_FRAGBUF                     =00660080       364     (1)
DS$_ICBUSY                      =006600C8       364     (1)
DS$_ICERR                       =006600C0       364     (1)
DS$_IHWE                        =00660060       364     (1)
DS$_ILLCHAR                     =00660018       364     (1)
DS$_ILLPAGCNT                   =00660078       364     (1)
DS$_ILLUNIT                     =00660100       364     (1)
DS$_INSFMEM                     =00660050       364     (1)
DS$_IPL2HI                      =006600B8       364     (1)
DS$_IVADDR                      =00660040       364     (1)
DS$_IVVECT                      =00660038       364     (1)
DS$_KRNLSTK                     =00660090       364     (1)
DS$_LOGIC                       =00660070       364     (1)
DS$_MCHK                        =00660088       364     (1)
DS$_MMOFF                       =00660058       364     (1)
DS$_NEEDUNIT                    =006600F8       364     (1)
DS$_NODE                        =00660128       364     (1)
DS$_NOPCS                       =00660110       364     (1)
DS$_NORMAL                      =00660001       364     (1)
DS$_NOSUPPORT                   =006600B1       364     (1)
DS$_NOTDON                      =00660030       364     (1)
DS$_NOTIMP                      =006600B0       364     (1)     364     (1)
DS$_NULLSTR                     =00660010       364     (1)
DS$_OVERFLOW                    =00660008       364     (1)
DS$_POWER                       =00660098       364     (1)
DS$_PROGERR                     =00660020       364     (1)
DS$_SEVERE                      =00660004       364     (1)
DS$_TRANSL                      =006600A0       364     (1)
DS$_TRUNCATE                    =00660028       364     (1)
DS$_UNEXPINT                    =006600D8       364     (1)
DS$_VASFULL                     =00660048       364     (1)
DS$_WARNING                     =00660000       364     (1)     364     (1)
DSA$GL_APTCOM                    0000FE04               #-2472  (6)
DSA$GL_FLAGS                     0000FE00                2468   (6)     2486    (6)     2487    (6)     2488    (6)
                                                         2957   (7)     2996    (7)     3036    (7)     3063    (7)
                                                         3064   (7)     3065    (7)     3072    (7)     3122    (7)
                                                         3124   (7)     3126    (7)     3128    (7)  #-3187     (7)
                                                         3450   (7)     3456    (7)     3463    (7)     3964    (10)
```

```
                                                    3979  (10)     4089  (10)     4108  (10)     4126  (10)
                                                    4197  (11)
DSA$V_APT                       =0000001F         #-2468  (6)
DSA$V_BELL                      =00000003         #-3493  (8)
DSA$V_BINARY                    =00000001         #-3503  (10)
DSA$V_CRD_AUTOTEST_OFF          =00000008         #-2486  (6)     #-3065  (7)     #-3128  (7)
DSA$V_CRD_MENUTEST_OFF          =00000010         #-2487  (6)     #-3063  (7)     #-3124  (7)     #-3126  (7)
DSA$V_CRD_MENUTEST_ON           =00000012         #-2488  (6)     #-3064  (7)
DSA$V_CRD_TRACE                 =00000011         #-3449  (7)     #-3455  (7)     #-3462  (7)
DSA$V_HALT                      =00000000         #-3511  (10)
DSA$V_IE1                       =00000004         #-3519  (10)
DSA$V_IE2                       =00000005         #-3527  (10)
DSA$V_IE3                       =00000006         #-3535  (10)
DSA$V_IES                       =00000007         #-3543  (10)
DSA$V_LOAD_DEBUGGER             =0000001E         #-3186  (7)
DSA$V_LOOP                      =00000002         #-3553  (10)
DSA$V_OPER                      =0000000C         #-3561  (10)
DSA$V_PROMPT                    =0000000D         #-3569  (10)
DSA$V_QUICK                     =00000008         #-3577  (10)
DSA$V_SEARCH                    =0000000E         #-2996  (7)     #-3036  (7)     #-3587  (10)
DSA$V_TRACE                     =0000000A         #-3595  (10)
DSA$V_USER                      =0000001C         #-2957  (7)     #-3072  (7)     #-3122  (7)     #-3963  (10)
                                                  #-3978  (10)    #-4089  (10)    #-4108  (10)    #-4126  (10)
                                                  #-4197  (11)
DSA$V_VERIFY                    =00000009         #-3603  (10)
DSR$COMPLETION                   00000000-XR        2946  (7)
DSR$LOAD_CRD                     00000000-XR        3085  (7)
DSR$UNLOAD_CRD                   00000000-XR        3142  (7)
DSV$ATTACH                       00000000-XR        2818  (7)
DSV$DEATTACH                     00000000-XR        2861  (7)
DSV$DEBUG                        00000000-XR        2870  (7)
DSV$DESELECT                     00000000-XR        2894  (7)
DSV$DIRECTORY                    00000000-XR        2902  (7)
DSV$EXIT                         00000000-XR        2925  (7)
DSV$INITPCS                      00000000-XR        2964  (7)
DSV$LOAD                         00000000-XR        2971  (7)
DSV$RUN                          00000000-XR        2993  (7)
DSV$SELECT                       00000000-XR        3006  (7)
DSV$SETLOAD                      00000000-XR        4070  (10)
DSV$SETPAGE                      00000000-XR        361?  (10)
DSV$SHOWDEVICE                   00000000-XR        3998  (10)    4010  (10)
DSV$SHOWDEVICEB                  00000000-XR        4001  (10)    4013  (10)
DSV$SHOWLOAD                     00000000-XR        4077  (10)
DSV$SHOWMEMORY                   00000000-XR        4171  (11)
DSV$SHOWPAGE                     00000000-XR        3620  (10)
DSV$SHOWSELECT                   00000000-XR        4015  (10)    4051  (10)
DSV$SHOWSELECTB                  00000000-XR        4018  (10)    4054  (10)
DSX$HELP                         00000000-XR        2943  (7)
DSX$MMOFF                        00000000-XR        4116  (10)
DSX$MMON                         00000000-XR        4097  (10)
DSX$PRINT                        00000000-XR        2480  (6)     2551  (6)     2562  (6)     2850  (7)
                                                    2961  (7)     3109  (7)     3138  (7)     3470  (7)
                                                    3732  (10)    4093  (10)    4112  (10)    4129  (10)
                                                    4140  (10)    4202  (11)
DSX$SUPERPARSE                   00000000-XR        2528  (6)
DS_CLEANUP                       00000000-XR        2936  (7)     3070  (7)
DS_ERRSUP                        00000000-XR        2740  (6)
```

```
DS_SUPERLIN"                    00000000-XR
EFN                             =0000004F       2685    (6)
END_CLEAR_FLAGS_LOOP            00000A74-R      1697    (5)

END_SET_FLAGS_LOOP              00000D7B-R      2000    (5)

ENUF                            =00000015       2627    (6)

EOL                             00000349-R      692     (1)

EVENT                           =00000034       2658    (6)
EXAMINE                         =00000010       2622    (6)
EXAMINE_PARAMS                  00000D87-R      2024    (5)

EXIT                            =00000011       2623    (6)
FAILURE                         =00000005       2611    (6)
FALSE                           =00000000       371     (1)

2500 (6)
1607 (5)  1812 (5)
1639 (5)  1644 (5)  1647 (5)  1650 (5)
1655 (5)  1658 (5)  1661 (5)  1664 (5)
1667 (5)  1671 (5)  1674 (5)  1677 (5)
1680 (5)  1683 (5)  1686 (5)
1943 (5)  1948 (5)  1951 (5)  1954 (5)
1957 (5)  1962 (5)  1965 (5)  1968 (5)
1971 (5)  1976 (5)  1979 (5)  1982 (5)
1985 (5)  1988 (5)  1991 (5)  1994 (5)
1029 (4)  1233 (4)  1240 (4)  1249 (4)
1253 (4)  1267 (4)  1292 (4)  1295 (4)
1306 (4)  1307 (4)  1308 (4)  1315 (4)
1322 (4)  1332 (4)  1334 (4)  1336 (4)
1338 (4)  1345 (4)  1354 (4)  1363 (4)
1370 (4)  1377 (4)  1384 (4)  1391 (4)
1398 (4)  1413 (4)  1422 (4)  1437 (5)
1439 (5)  1448 (5)  1455 (5)  1461 (5)
1512 (5)  1550 (5)  1555 (5)  1572 (5)
1583 (5)  1596 (5)  1605 (5)  1702 (5)
1744 (5)  1753 (5)  1770 (5)  1781 (5)
1801 (5)  1810 (5)  1848 (5)  1859 (5)
1862 (5)  1872 (5)  1894 (5)  1897 (5)
1902 (5)  1907 (5)  1912 (5)  1917 (5)
1926 (5)  2005 (5)  2168 (6)  2330 (6)
2404 (6)  454 (1)   461 (1)   486 (1)
541 (1)   571 (1)   597 (1)   599 (1)
606 (1)   623 (1)   659 (1)   666 (1)
674 (1)   849 (4)   852 (4)   985 (4)
1029 (4)  1233 (4)  1240 (4)  1249 (4)
1253 (4)  1267 (4)  1297 (4)  1306 (4)
1307 (4)  1308 (4)  1315 (4)  1322 (4)
1330 (4)  1345 (4)  1354 (4)  1363 (4)
1370 (4)  1377 (4)  1384 (4)  1391 (4)
1398 (4)  1422 (4)  1437 (5)  1439 (5)
1448 (5)  1455 (5)  1461 (5)  1512 (5)
1550 (5)  1555 (5)  1572 (5)  1583 (5)
1596 (5)  1605 (5)  1702 (5)  1744 (5)
1753 (5)  1770 (5)  1781 (5)  1801 (5)
1810 (5)  1848 (5)  1859 (5)  1862 (5)
1872 (5)  1894 (5)  1897 (5)  1902 (5)
1907 (5)  1912 (5)  1917 (5)  1926 (5)
2005 (5)  2168 (6)  2330 (6)  2404 (6)
454 (1)   486 (1)   515 (1)   541 (1)
577 (1)   606 (1)   621 (1)   623 (1)
666 (1)   674 (1)   680 (1)   849 (4)
852 (4)   878 (4)   985 (4)
1305 (4)  1597 (5)  1803 (5)
587 (1)
2032 (5)  2039 (5)  2046 (5)  2053 (5)
2060 (5)  2069 (5)  2076 (5)  2083 (5)
2104 (6)  2115 (6)  2125 (6)  2132 (6)
2139 (6)  2147 (6)  2155 (6)  2162 (6)
588 (1)
577 (1)
```

FILEND	=00000025	2643	(6)	1046	(4)	1047	(4)	1049	(4)	1076	(4)		
				1079	(4)	1081	(4)	1083	(4)	1084	(4)		
				1086	(4)	1089	(4)	1090	(4)	1092	(4)		
				1093	(4)	1095	(4)	1098	(4)	1099	(4)		
				840	(4)	841	(4)	872	(4)	873	(4)		
				916	(4)	917	(4)	919	(4)	946	(4)		
				949	(4)	951	(4)	953	(4)	954	(4)		
				956	(4)	959	(4)	960	(4)	962	(4)		
				963	(4)	965	(4)	968	(4)	969	(4)		
FILESPEC	=00000024	2642	(6)	1044	(4)	838	(4)	871	(4)	914	(4)		
FLAGS	=00000035	2659	(6)	1314	(4)	1600	(5)	1618	(5)	1805	(5)		
				1820	(5)								
GET_DEV_NAME	000003B0-R	835	(4)	1298	(4)	1509	(5)	902	(4)				
GET_FILE_SPEC	00000450-R	913	(4)	1833	(5)	516	(1)	539	(1)	614	(1)		
				683	(1)								
GT_MENU_ERROR	000000F3-R	424	(1)	3109	(7)								
HALT	=00000C3D	2667	(6)	1649	(5)	1956	(5)						
HELP	=00000012	2624	(6)	596	(1)	598	(1)						
HELPINFO	00000000-XR			2937	(7)	#-2941	(7)						
HEX	=00000061	2703	(6)	2052	(5)	2205	(6)	2368	(6)				
IE1	=0000003E	2668	(6)	1654	(5)	1961	(5)						
IE2	=0000003F	2669	(6)	1657	(5)	1964	(5)						
IE3	=00000040	2670	(6)	1660	(5)	1967	(5)						
IE5	=00000041	2671	(6)	1663	(5)	1970	(5)						
IFNOTUSER	=00000069	2711	(6)										
IF_ADAPTER	=00000067	2709	(6)	876	(4)	878	(4)						
IF_FILE_SPEC	=00000066	2708	(6)	1101	(4)	869	(4)	982	(4)				
IF_REG_OR_ADR	=00000068	2710	(6)	2091	(5)	2245	(6)						
IF_REQUIRED	=00000065	2707	(6)	851	(4)	983	(4)						
IF_RUN	=00000064	2706	(6)	1027	(4)								
IF_XDELTA	=0000006A	2712	(6)	672	(1)								
ILLEGAL	00000361-R	712	(4)	1042	(4)	1128	(4)	1129	(4)	1146	(4)		
				1147	(4)	1154	(4)	1155	(4)	1168	(4)		
				1169	(4)	1171	(4)	1239	(4)	1248	(4)		
				1251	(4)	1252	(4)	1260	(4)	1281	(4)		
				1286	(4)	1289	(4)	1311	(4)	1353	(4)		
				1357	(4)	1397	(4)	1433	(5)	1454	(5)		
				1482	(5)	1489	(5)	1492	(5)	1506	(5)		
				1511	(5)	1535	(5)	1546	(5)	1554	(5)		
				1571	(5)	1582	(5)	1595	(5)	1597	(5)		
				1598	(5)	1601	(5)	1619	(5)	1733	(5)		
				1742	(5)	1743	(5)	1751	(5)	1752	(5)		
				1769	(5)	1780	(5)	1800	(5)	1803	(5)		
				1804	(5)	1806	(5)	1821	(5)	1830	(5)		
				1832	(5)	1846	(5)	1847	(5)	1856	(5)		
				1857	(5)	1861	(5)	1869	(5)	1870	(5)		
				1892	(5)	1893	(5)	1900	(5)	1901	(5)		
				1905	(5)	1906	(5)	1910	(5)	1911	(5)		
				1914	(5)	1915	(5)	1916	(5)	1924	(5)		
				1925	(5)	2067	(5)	2068	(5)	2094	(5)		
				2138	(6)	2146	(6)	2154	(6)	2161	(6)		
				2220	(6)	2221	(6)	2248	(6)	2292	(6)		
				2301	(6)	2309	(6)	2316	(6)	2322	(6)		
				2323	(6)	2348	(6)	460	(1)	492	(1)		
				493	(1)	530	(1)	537	(1)	547	(1)		
				558	(1)	587	(1)	598	(1)	613	(1)		
				622	(1)	645	(1)	665	(1)	863	(4)		

				866	(4)	870	(4)	895	(4)	898	(4)
				901	(4)						
INC_CMD	=00000001	2607	(6)	702	(4)	752	(4)	794	(4)	823	(4)
INCOMPLETE	00000379-R	759	(4)	1055	(4)	1068	(4)	1079	(4)	1103	(4)
				783	(4)	812	(4)	851	(4)	872	(4)
				879	(4)	925	(4)	938	(4)	949	(4)
				983	(4)						
INC_CMD	=00000002	2608	(6)	731	(4)	764	(4)	785	(4)	814	(4)
INISBRK	00000000-XR			3984	(10)						
INITPCS	=0000007F	2733	(6)	605	(1)						
INIT_CONTEXT	00000000-XR			2940	(7)						
KB_CHECK	00000000-XR			2491	(6)						
KB_CHECK_APT	00000000-XR			2471	(6)						
LAST	=00000029	2647	(6)	1171	(4)						
LINE_COUNT	00000C00-XR			#-2502	(6)						
LOAD	=00000013	2625	(6)	612	(1)						
LONG	=0000005E	2700	(6)	2059	(5)	2211	(6)	2375	(6)		
LOOP	=00000042	2672	(6)	1666	(5)	1974	(5)				
MAPFREE	00000000-XR			3073	(7)						
MMOFF	=0000002C	2650	(6)	1861	(5)						
MMON	=0000002B	2649	(6)	1858	(5)						
NEXT	=00000050	2686	(6)	2068	(5)	2221	(6)				
NEXT_INST	=00000014	2626	(6)	620	(1)						
NOTNUF	=00000016	2628	(6)	1025	(4)	1127	(4)	1128	(4)	1143	(4)
				1145	(4)	1146	(4)	1153	(4)	1154	(4)
				1167	(4)	1168	(4)	1170	(4)	1226	(4)
				1246	(4)	1259	(4)	1260	(4)	1283	(4)
				1286	(4)	1330	(4)	1352	(4)	1356	(4)
				1357	(4)	1401	(4)	1412	(4)	1434	(5)
				1482	(5)	1489	(5)	1499	(5)	1506	(5)
				1543	(5)	1548	(5)	1568	(5)	1579	(5)
				1580	(5)	1608	(5)	1619	(5)	1620	(5)
				1626	(5)	1641	(5)	1652	(5)	1653	(5)
				1699	(5)	1735	(5)	1742	(5)	1751	(5)
				1766	(5)	1777	(5)	1778	(5)	1788	(5)
				1813	(5)	1825	(5)	1829	(5)	1830	(5)
				1832	(5)	1845	(5)	1855	(5)	1857	(5)
				1868	(5)	1869	(5)	1884	(5)	1885	(5)
				1892	(5)	1900	(5)	1905	(5)	1910	(5)
				1915	(5)	1924	(5)	1945	(5)	1959	(5)
				1960	(5)	1973	(5)	1974	(5)	2002	(5)
				2025	(5)	2067	(5)	2094	(5)	2185	(6)
				2220	(6)	2248	(6)	2397	(6)	447	(1)
				472	(1)	479	(1)	506	(1)	513	(1)
				531	(1)	548	(1)	559	(1)	569	(1)
				570	(1)	588	(1)	613	(1)	630	(1)
				637	(1)	638	(1)	640	(1)	646	(1)
				652	(1)	681	(1)	860	(4)	863	(4)
				877	(4)	892	(4)	895	(4)		
NULL	=00000000	2606	(6)								
OCT	=00000063	2705	(6)	2075	(5)	2228	(6)	2382	(6)		
OFF	=00000000	371	(1)								
ON	=00000001	371	(1)								
OPER	=00000043	2673	(6)	1670	(5)	1978	(5)				
OPTIONAL	=00000026	2644	(6)	1298	(4)	1509	(5)	516	(1)	539	(1)
PAGE	=00000051	2687	(6)	1353	(4)	1871	(5)				
PASS	=00000030	2654	(6)	1129	(4)						

```
PREGN                           =0000002E       2652    (6)       2138    (6)       2292    (6)
PROMPT                          =00000044       2674    (6)       1673    (5)       1981    (5)
QA                              =00000031       2655    (6)       1136    (4)
QACL                            =0000004A       2680    (6)       1362    (4)       1891    (5)
QADEF                           =0000004E       2684    (6)       1369    (4)       1896    (5)
QAEP                            =00000049       2679    (6)       1376    (4)       1899    (5)
QAMP                            =0000004D       2683    (6)       1383    (4)       1904    (5)
QASL                            =0000004C       2682    (6)       1390    (4)       1909    (5)
QATL                            =0000004B       2681    (6)       1397    (4)       1914    (5)
QA_COMMON                        0000074D-R     3674    (10)    #-3651    (10)    #-3652    (10)    #-3656    (10)    #-3657    (10)
                                                                 #-3661    (10)    #-3662    (10)    #-3666    (10)    #-3667    (10)
                                                                 #-3671    (10)    #-3695    (10)
QUICK                           =00000045       2675    (6)       1676    (5)       1984    (5)
Q_ADAPTER                        00000000-XR                     #-2451    (6)     #-3932    (10)    #-4028    (10)    #-4029    (10)
                                                                 #-4036    (10)    #-4037    (10)
Q_GOOD_CMD                       00000444-R     383     (1)       2547    (6)       2558    (6)     #-2775    (7)     #-2776    (7)
                                                                 #-2777    (7)
RCLI                             00000002-R     2450    (6)     #-2482    (6)     #-2509    (6)     #-2514    (6)     #-2552    (6)
                                                                 #-2563    (6)     #-2572    (6)
REG12                           =00000055       2691    (6)       2102    (6)       2255    (6)
REG13                           =00000056       2692    (6)       2113    (6)       2266    (6)
REG14                           =00000057       2693    (6)       2153    (6)       2308    (6)
REG15                           =00000058       2694    (6)       2124    (6)       2278    (6)
REG16                           =00000059       2695    (6)       2131    (6)       2285    (6)
REGN                            =00000054       2690    (6)       2145    (6)       2300    (6)
REGP                            =0000005A       2696    (6)       2103    (6)       2114    (6)       2154    (6)       2256    (6)
                                                                    2267    (6)       2309    (6)
REQUIRED                        =00000027       2645    (6)       1833    (5)       614     (1)       683     (1)       902     (4)
RUN                             =00000017       2629    (6)       629     (1)
SCRIPT                          =00000018       2630    (6)       680     (1)
SCRIPT$CONT                      00000000-XR                      2851    (7)
SCRIPT$FLUSH                     00000000-XR                      2935    (7)
SCRIPT$OPEN                      00000000-XR                      3044    (7)
SEARCH                          =00000046       2676    (6)       1679    (5)       1987    (5)
SECTION                         =0000002F       2653    (6)       1147    (4)
SELECT                          =00000071       2719    (6)       645     (1)
SELECT_QUALS                     00000435-R     890     (3)       646     (1)
SET                             =00000019       2631    (6)       639     (1)
SETENFORCE                      =00000032       2656    (6)       1800    (5)
SETLOAD                         =00000074       2722    (6)       1831    (5)
SETMEM                          =0000007E       2732    (6)       1845    (5)
SET_CRD_DEBUG                   =0000001D       2635    (6)       1780    (5)
SET_CRD_TRACE                   =0000001C       2634    (6)       1769    (5)
SET_DEFAULT_PARAMS               00000F5F-R     2347    (6)       1788    (5)
SET_FLAGS_LOOP                   00000C9F-R     1940    (5)       1755    (5)       1823    (5)       1837    (5)       1888    (5)
                                                                    1923    (5)       2003    (5)
SET_PARAMS                       00000A80-R     1732    (5)       640     (1)
SHOMEMALL                       =0000007D       2731    (6)       1337    (4)
SHOMEMBUF                       =0000007B       2729    (6)       1333    (4)
SHOMEMDAT                       =0000007C       2730    (6)       1335    (4)
SHOMEMMAP                       =0000007A       2728    (6)       1331    (4)
SHOW                            =0000001A       2632    (6)       651     (1)
SHOWLOAD                        =00000075       2723    (6)       1321    (4)
SHOWMEM                         =00000079       2727    (6)       1329    (4)
SHOWMEMORYFLAGS                  00000000-XR                    #-2452    (6)       4185    (11)
SHOWNM                          =0000002D       2651    (6)       1344    (4)
SHOWSECTIONS                    =00000078       2726    (6)       1421    (4)
```

```
SHOWSEL                         =00000072      2720  (6)       1433  (5)       1511  (5)
SHOWSTATUS                      =00000077      2725  (6)       1447  (5)
SHOWSUP                         =00000076      2724  (6)       1454  (5)
SHOW_CALLS                      =00000020      2638  (6)       1252  (4)
SHOW_CRD_TRACE                  =0000001F      2637  (6)       1248  (4)
SHOW_PARAMS                      00000648-R    1223  (4)        652  (1)
SIZ...                          =00000001       366  (1)        366  (1)
SS$_ENDOFFILE                    00000000-XR                  #-2504  (6)
START                           =00000021      2639  (6)        658  (1)
START_AND_RUN                    00000509-R    1022  (4)       1101  (4)       1120  (4)       1130  (4)       1137  (4)
                                                               1148  (4)       1156  (4)       1172  (4)       1192  (4)
                                                                630  (1)        659  (1)
START_RUN_QUALIFIERS             000005C5-R    1113  (4)       1025  (4)
SUBTEST                         =0000002A      2648  (6)       1155  (4)
SUCCESS                         =00000C04      2610  (6)
SUMMARY                         =00000022      2640  (6)        665  (1)
SYS$EXIT                         00000000-XR                   2506  (6)
TEST                            =00000028      2646  (6)       1169  (4)
TRACE                           =00000047      2677  (6)       1682  (5)       1990  (5)
TRUE                            =00000001       371  (1)
T_CONT                           00000092-R     409  (1)       2850  (7)
T_CRD_TRACE_OUTPUT               00000163-R     430  (1)       3470  (7)
T_EMESSG1                        000000AB-R     412  (1)       2958  (7)       4090  (10)      4109  (10)      4129  (10)
                                                               4199  (11)
T_ILLCMD                         00000018-R     400  (1)       2548  (6)
T_ILLFN                          00000061-R     406  (1)       3729  (10)
T_INCOMPLETE                     0000003A-R     403  (1)       2559  (6)
T_OFF                            000000EF-R     421  (1)       3465  (7)       4132  (10)
T_ON                             000000EC-R     418  (1)       3461  (7)       4135  (10)
T_PRGERR                         00000004-R     397  (1)       2740  (6)
T_SHOWMM                         000000D0-R     415  (1)       4137  (10)
T_USER_MODE_MENU_ERROR           00000129-R     427  (1)       3138  (7)
VERIFY                          =00000048      2678  (6)       1685  (5)       1993  (5)
VRABORT                          00000000-XR                   2825  (7)
VRCLRBRK                         00000000-XR                   3383  (7)
VRCLREF                          00000000-XR                   3325  (7)
VRCLRFLC                         00000000-XR                   2833  (7)       3405  (7)
VRDEPOSIT                        00000000-XR                   2884  (7)
VREXAMINE                        00000000-XR                   2915  (7)
VRSETBASE                        00000000-XR                   3357  (7)
VRSETBRK                         00000000-XR                   3378  (7)
VRSETDFLT                        00000000-XR                   3422  (7)
VRSETEF                          00000000-XR                   3320  (7)
VRSETFLG                         00000000-XR                   3013  (7)       3402  (7)
VRSETMEM                         00000000-XR                   4206  (11)
VRSETQA                          00000000-XR                   3680  (10)      3705  (10)
VRSETWIDTH                       00000000-XR                   3634  (10)
VRSHOWBASE                       00000000-XR                   3354  (7)
VRSHOWBRK                        00000000-XR                   3386  (7)
VRSHOWDFLT                       00000000-XR                   3418  (7)
VRSHOWEF                         00000000-XR                   3328  (7)
VRSHOWFLG                        00000000-XR                   3342  (7)
VRSHOWQA                         00000000-XR                   3684  (10)      3710  (10)
VRSHOWSECTIONS                   00000000-XR                   4164  (11)
VRSHOWSTATUS                     00000000-XR                   4149  (10)
VRSHOWWIDTH                      00000000-XR                   3637  (10)
VRSHOW_CALLS                     00000000-XR                   3437  (7)
```

```
VRSTART                 00000000-XR                     3033    (7)
VRSUMMARY               00000000-XR                     3052    (7)
VRSUPPORT               00000000-XR                     4157    (11)
WIDE_FLAG               00000000-XR                   #-2904    (7)     #-2908    (7)
WIDTH                   =00000052       2688    (6)     1460    (5)       1923    (5)
WORD                    =0000005F       2701    (6)     2082    (5)       2235    (6)       2389    (6)
XDELBPT                 00000000-XR                     3974    (10)    #-3976    (10)
XDELTA                  =00000068       2713    (6)     673     (1)
XDELTBIT                00000000-XR                     3974    (10)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CRD_LITERALS	2	371 (1)	371 (1)
$D1_PRINT_S	1	2480 (6)	2480 (6) 2850 (7) 3109 (7) 3138 (7) 3470 (7) 4129 (10)
$DEF	1	371 (1)	
$DEFINI	1	363 (1)	363 (1) 367 (1) 368 (1) 369 (1)
$DS_CLI	1	445 (1)	1024 (4) 1025 (4) 1027 (4) 1029 (4) 1042 (4) 1044 (4) 1046 (4) 1047 (4) 1049 (4) 1051 (4) 1052 (4) 1054 (4) 1055 (4) 1057 (4) 1059 (4) 1061 (4) 1062 (4) 1063 (4) 1064 (4) 1065 (4) 1067 (4) 1068 (4) 1070 (4) 1071 (4) 1072 (4) 1073 (4) 1074 (4) 1076 (4) 1077 (4) 1079 (4) 1081 (4) 1083 (4) 1084 (4) 1085 (4) 1086 (4) 1087 (4) 1089 (4) 1090 (4) 1092 (4) 1093 (4) 1094 (4) 1095 (4) 1096 (4) 1098 (4) 1099 (4) 1101 (4) 1103 (4) 1119 (4) 1120 (4) 1127 (4) 1128 (4) 1129 (4) 1130 (4) 1136 (4) 1137 (4) 1143 (4) 1145 (4) 1146 (4) 1147 (4) 1148 (4) 1153 (4) 1154 (4) 1155 (4) 1156 (4) 1167 (4) 1168 (4) 1169 (4) 1170 (4) 1171 (4) 1172 (4) 1192 (4) 1224 (4) 1226 (4) 1232 (4) 1233 (4) 1239 (4) 1240 (4) 1246 (4) 1247 (4) 1248 (4) 1249 (4) 1251 (4) 1252 (4) 1253 (4) 1259 (4) 1260 (4) 1266 (4) 1267 (4) 1281 (4) 1283 (4) 1285 (4) 1286 (4) 1287 (4) 1289 (4) 1290 (4) 1292 (4) 1294 (4) 1295 (4) 1297 (4) 1298 (4) 1305 (4) 1306 (4) 1307 (4) 1308 (4) 1314 (4) 1315 (4) 1321 (4) 1322 (4) 1328 (4) 1329 (4) 1330 (4) 1331 (4) 1332 (4) 1333 (4) 1334 (4) 1335 (4) 1336 (4) 1337 (4) 1338 (4) 1344 (4) 1345 (4) 1352 (4) 1353 (4) 1354 (4) 1356 (4) 1357 (4) 1362 (4) 1363 (4) 1369 (4) 1370 (4) 1376 (4) 1377 (4) 1383 (4) 1384 (4) 1390 (4) 1391 (4) 1397 (4) 1398 (4) 1401 (4) 1412 (4) 1413 (4) 1419 (4) 1421 (4) 1422 (4) 1431 (5) 1433 (5) 1434 (5) 1436 (5) 1437 (5) 1439 (5) 1447 (5) 1448 (5) 1454 (5) 1455 (5) 1460 (5) 1461 (5) 1482 (5) 1488 (5) 1489 (5) 1490 (5) 1492 (5) 1493 (5) 1495 (5) 1497 (5) 1499 (5) 1500 (5) 1506 (5) 1508 (5) 1509 (5) 1511 (5) 1512 (5) 1535 (5) 1543 (5) 1545 (5) 1546 (5) 1548 (5) 1549 (5) 1550 (5) 1552 (5) 1554 (5)

1555 (5) 1557 (5) 1567 (5) 1568 (5)
1569 (5) 1571 (5) 1572 (5) 1579 (5)
1580 (5) 1581 (5) 1582 (5) 1583 (5)
1593 (5) 1594 (5) 1595 (5) 1596 (5)
1597 (5) 1598 (5) 1600 (5) 1601 (5)
1603 (5) 1605 (5) 1607 (5) 1608 (5)
1610 (5) 1618 (5) 1619 (5) 1620 (5)
1626 (5) 1638 (5) 1639 (5) 1641 (5)
1643 (5) 1644 (5) 1646 (5) 1647 (5)
1649 (5) 1650 (5) 1652 (5) 1653 (5)
1654 (5) 1655 (5) 1657 (5) 1658 (5)
1660 (5) 1661 (5) 1663 (5) 1664 (5)
1666 (5) 1667 (5) 1670 (5) 1671 (5)
1673 (5) 1674 (5) 1676 (5) 1677 (5)
1679 (5) 1680 (5) 1682 (5) 1683 (5)
1685 (5) 1686 (5) 1699 (5) 1700 (5)
1702 (5) 1733 (5) 1735 (5) 1741 (5)
1742 (5) 1743 (5) 1744 (5) 1750 (5)
1751 (5) 1752 (5) 1753 (5) 1755 (5)
1765 (5) 1766 (5) 1767 (5) 1769 (5)
1770 (5) 1777 (5) 1778 (5) 1779 (5)
1780 (5) 1781 (5) 1787 (5) 1788 (5)
1798 (5) 1799 (5) 1800 (5) 1801 (5)
1803 (5) 1804 (5) 1805 (5) 1806 (5)
1808 (5) 1810 (5) 1812 (5) 1813 (5)
1814 (5) 1820 (5) 1821 (5) 1823 (5)
1829 (5) 1830 (5) 1831 (5) 1832 (5)
1833 (5) 1835 (5) 1837 (5) 1843 (5)
1845 (5) 1846 (5) 1847 (5) 1848 (5)
1855 (5) 1856 (5) 1857 (5) 1858 (5)
1859 (5) 1861 (5) 1862 (5) 1868 (5)
1869 (5) 1870 (5) 1871 (5) 1872 (5)
1884 (5) 1885 (5) 1886 (5) 1888 (5)
1891 (5) 1892 (5) 1893 (5) 1894 (5)
1896 (5) 1897 (5) 1899 (5) 1900 (5)
1901 (5) 1902 (5) 1904 (5) 1905 (5)
1906 (5) 1907 (5) 1909 (5) 1910 (5)
1911 (5) 1912 (5) 1914 (5) 1915 (5)
1916 (5) 1917 (5) 1923 (5) 1924 (5)
1925 (5) 1926 (5) 1942 (5) 1943 (5)
1945 (5) 1947 (5) 1949 (5) 1950 (5)
1951 (5) 1953 (5) 1954 (5) 1956 (5)
1957 (5) 1959 (5) 1960 (5) 1961 (5)
1962 (5) 1964 (5) 1965 (5) 1967 (5)
1968 (5) 1970 (5) 1971 (5) 1973 (5)
1974 (5) 1975 (5) 1976 (5) 1978 (5)
1979 (5) 1981 (5) 1982 (5) 1984 (5)
1985 (5) 1987 (5) 1988 (5) 1990 (5)
1991 (5) 1993 (5) 1994 (5) 2002 (5)
2003 (5) 2005 (5) 2025 (5) 2031 (5)
2032 (5) 2038 (5) 2039 (5) 2045 (5)
2046 (5) 2052 (5) 2053 (5) 2059 (5)
2060 (5) 2066 (5) 2067 (5) 2068 (5)
2069 (5) 2075 (5) 2076 (5) 2082 (5)
2083 (5) 2091 (5) 2094 (5) 2102 (6)
2103 (6) 2104 (6) 2106 (6) 2113 (6)
2114 (6) 2115 (6) 2117 (6) 2123 (6)

2124 (6) 2125 (6) 2131 (6) 2132 (6)
2138 (6) 2139 (6) 2145 (6) 2146 (6)
2147 (6) 2153 (6) 2154 (6) 2155 (6)
2161 (6) 2162 (6) 2168 (6) 2185 (6)
2191 (6) 2192 (6) 2198 (6) 2199 (6)
2205 (6) 2206 (6) 2212 (6) 2213 (6)
2219 (6) 2220 (6) 2221 (6) 2222 (6)
2228 (6) 2229 (6) 2235 (6) 2236 (6)
2245 (6) 2248 (6) 2255 (6) 2256 (6)
2257 (6) 2259 (6) 2266 (6) 2267 (6)
2268 (6) 2270 (6) 2277 (6) 2278 (6)
2279 (6) 2285 (6) 2286 (6) 2292 (6)
2293 (6) 2300 (6) 2301 (6) 2302 (6)
2308 (6) 2309 (6) 2310 (6) 2316 (6)
2322 (6) 2323 (6) 2324 (6) 2330 (6)
2348 (6) 2354 (6) 2355 (6) 2361 (6)
2362 (6) 2368 (6) 2369 (6) 2375 (6)
2376 (6) 2382 (6) 2383 (6) 2389 (6)
2390 (6) 2397 (6) 2398 (6) 2404 (6)
445 (1) 447 (1) 453 (1) 454 (1)
460 (1) 461 (1) 472 (1) 478 (1)
479 (1) 485 (1) 486 (1) 492 (1)
493 (1) 506 (1) 512 (1) 513 (1)
514 (1) 515 (1) 516 (1) 518 (1)
519 (1) 520 (1) 521 (1) 522 (1)
523 (1) 524 (1) 530 (1) 531 (1)
537 (1) 538 (1) 539 (1) 540 (1)
541 (1) 547 (1) 548 (1) 550 (1)
552 (1) 558 (1) 559 (1) 569 (1)
570 (1) 571 (1) 577 (1) 579 (1)
581 (1) 587 (1) 588 (1) 595 (1)
596 (1) 597 (1) 598 (1) 599 (1)
605 (1) 606 (1) 612 (1) 613 (1)
614 (1) 620 (1) 621 (1) 622 (1)
623 (1) 629 (1) 630 (1) 637 (1)
638 (1) 639 (1) 640 (1) 645 (1)
646 (1) 651 (1) 652 (1) 658 (1)
659 (1) 665 (1) 666 (1) 672 (1)
673 (1) 674 (1) 680 (1) 681 (1)
683 (1) 692 (1) 693 (1) 696 (2)
699 (4) 702 (4) 705 (4) 731 (4)
735 (4) 736 (4) 743 (4) 745 (4)
752 (4) 764 (4) 782 (4) 783 (4)
785 (4) 787 (4) 788 (4) 790 (4)
792 (4) 794 (4) 811 (4) 812 (4)
814 (4) 816 (4) 817 (4) 819 (4)
821 (4) 823 (4) 836 (4) 838 (4)
840 (4) 841 (4) 842 (4) 843 (4)
845 (4) 847 (4) 849 (4) 851 (4)
852 (4) 860 (4) 862 (4) 863 (4)
864 (4) 866 (4) 867 (4) 869 (4)
870 (4) 871 (4) 872 (4) 873 (4)
874 (4) 875 (4) 876 (4) 877 (4)
878 (4) 879 (4) 892 (4) 894 (4)
895 (4) 896 (4) 898 (4) 899 (4)
901 (4) 902 (4) 914 (4) 916 (4)
917 (4) 919 (4) 921 (4) 922 (4)

```
                                                                924     (4)     925     (4)     927     (4)     929     (4)
                                                                931     (4)     932     (4)     933     (4)     934     (4)
                                                                935     (4)     937     (4)     938     (4)     940     (4)
                                                                941     (4)     942     (4)     943     (4)     944     (4)
                                                                946     (4)     947     (4)     949     (4)     951     (4)
                                                                953     (4)     954     (4)     955     (4)     956     (4)
                                                                957     (4)     959     (4)     960     (4)     962     (4)
                                                                963     (4)     964     (4)     965     (4)     966     (4)
                                                                968     (4)     969     (4)     982     (4)     983     (4)
                                                                985     (4)
$DS_CLIDEF                  1           362     (1)             362     (1)
$DS_DSADEF                  5           363     (1)             363     (1)
$DS_DSDEF                   2           364     (1)             364     (1)
$DS_TYPEDEF                 4           370     (1)             370     (1)
$EQU                        1           371     (1)             362     (1)     364     (1)     370     (1)     371     (1)
$EQULS1                     1           371     (1)             362     (1)     364     (1)     370     (1)     371     (1)
$EQULST                     1           362     (1)             362     (1)     364     (1)     370     (1)     371     (1)
$EXIT_S                     1           2506    (6)             2506    (6)
$GBLINI                     2           362     (1)             362     (1)     364     (1)     366     (1)     370     (1)
                                                                371     (1)
$IPLDEF                     1           367     (1)             367     (1)
$PRDEF                      4           368     (1)             368     (1)
$PRINT                      2           2478    (6)             2478    (6)     2847    (7)     3107    (7)     3136    (7)
                                                                3467    (7)     4127    (10)
$PSLDEF                     2           369     (1)             369     (1)
$PUSHADR                    1           2740    (6)             2740    (6)
$VIELD                      1                                   366     (1)
$VIELD1                     1           371     (1)             366     (1)
BR_IF_CRD_AUTOTEST_OFF      1           2486    (6)             2486    (6)     3065    (7)
BR_IF_CRD_MENUTEST_OFF      1           2487    (6)             2487    (6)     3063    (7)
BR_IF_CRD_MENUTEST_ON       1           2488    (6)             2488    (6)     3064    (7)
BR_IF_NOT_APT               1           2468    (6)             2468    (6)
BR_IF_NOT_USER              1           2957    (7)             2957    (7)     4089    (10)    4108    (10)    4126    (10)
                                                                4197    (11)
BR_IF_USER                  1           3072    (7)             3072    (7)     3122    (7)
CASE                        1           3118    (7)             3118    (7)
CASEDEF                     1           352     (1)             2606    (6)     2607    (6)     2608    (6)     2609    (6)
                                                                2610    (6)     2611    (6)     2612    (6)     2613    (6)
                                                                2614    (6)     2615    (6)     2616    (6)     2617    (6)
                                                                2618    (6)     2619    (6)     2620    (6)     2621    (6)
                                                                2622    (6)     2623    (6)     2624    (6)     2625    (6)
                                                                2626    (6)     2627    (6)     2628    (6)     2629    (6)
                                                                2630    (6)     2631    (6)     2632    (6)     2633    (6)
                                                                2634    (6)     2635    (6)     2636    (6)     2637    (6)
                                                                2638    (6)     2639    (6)     2640    (6)     2641    (6)
                                                                2642    (6)     2643    (6)     2644    (6)     2645    (6)
                                                                2646    (6)     2647    (6)     2648    (6)     2649    (6)
                                                                2650    (6)     2651    (6)     2652    (6)     2653    (6)
                                                                2654    (6)     2655    (6)     2656    (6)     2657    (6)
                                                                2658    (6)     2659    (6)     2660    (6)     2661    (6)
                                                                2662    (6)     2663    (6)     2664    (6)     2665    (6)
                                                                2666    (6)     2667    (6)     2668    (6)     2669    (6)
                                                                2670    (6)     2671    (6)     2672    (6)     2673    (6)
                                                                2674    (6)     2675    (6)     2676    (6)     2677    (6)
                                                                2678    (6)     2679    (6)     2680    (6)     2681    (6)
                                                                2682    (6)     2683    (6)     2684    (6)     2685    (6)
                                                                2686    (6)     2687    (6)     2688    (6)     2689    (6)
```

				2690	(6)	2691	(6)	2692	(6)	2693	(6)
				2694	(6)	2695	(6)	2696	(6)	2697	(6)
				2698	(6)	2699	(6)	2700	(6)	2701	(6)
				2702	(6)	2703	(6)	2704	(6)	2705	(6)
				2706	(6)	2707	(6)	2708	(6)	2709	(6)
				2710	(6)	2711	(6)	2712	(6)	2713	(6)
				2714	(6)	2715	(6)	2716	(6)	2717	(6)
				2718	(6)	2719	(6)	2720	(6)	2721	(6)
				2722	(6)	2723	(6)	2724	(6)	2725	(6)
				2726	(6)	2727	(6)	2728	(6)	2729	(6)
				2730	(6)	2731	(6)	2732	(6)	2733	(6)
				2734	(6)						
CLEAR_CTRLC	1	245	(6)	2494	(6)						
CLEAR_CTRLO	1	247	(6)	2477	(6)						
CLIDEF	3	365	(1)	365	(1)						
DSFDEF	3	366	(1)	366	(1)						
ERRSUP_S	1	2740	(6)	2740	(6)						
MODNAM	1	395	(1)	395	(1)						
SET_CRD_AUTOTEST_OFF	1	3128	(7)	3128	(7)						
SET_CRD_MENUTEST_OFF	1	3124	(7)	3124	(7)	3126	(7)				

+-------------------------+
! Performance indicators !
+-------------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.23
Command processing	141	00:00:00.70	00:00:01.55
Pass 1	2292	00:01:04.46	00:01:22.80
Symbol table sort	2	00:00:01.87	00:00:02.27
Pass 2	1335	00:00:12.90	00:00:18.28
Symbol table output	82	00:00:00.52	00:00:00.87
Psect synopsis output	8	00:00:00.04	00:00:00.04
Cross-reference output	549	00:00:10.21	00:00:12.19
Assembler run totals	4446	00:01:30.82	00:01:58.24

The working set limit was 1000 pages.
345368 bytes (675 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 829 non-local and 433 local symbols.
4209 source lines were read in Pass 1, producing 0 object records in Pass 2.
113 pages of virtual memory were used to define 37 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	7
DRB1:[DS.WORK]DS.MLB;218	15
DRB1:[DS.WORK]CRD.MLB;12	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	33

673 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CLI/UPDA=(CLI.UPD,CLI.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DIAG/
```

B 1 $SETMAP CALL
C 1 $SETMAP CALL
D 1 DSP$SHOWMBA
E 1 DSP$SHOWUBI
F 1 $SHOWCHAN CALL
G 1 SHOW UBI STATUS
H 1 SHOWBITS
I 1 DETERMINE ADAPTER SPECIFIC STA
J 1 DETERMINE GENERAL ADAPTER STAT
K 1 DETERMINE ADAPTER HARDWARE ERR
L 1 DETERMINE ADAPTER HARDWARE ERR
M 1 BUILD THE CHANNEL DATA BLOCK
N 1 SET ADAPTER VECTOR(S)
B 2 CLEAR ADAPTER VECTOR(S)
C 2 INTERRUPT PRE-PROCESS
D 2 INTERRUPT PRE-PROCESS
E 2 Symbol table
F 2 Symbol table
G 2 Psect synopsis
H 2 Cross reference
I 2 Cross reference
J 2 Cross reference
K 2 Cross reference
L 2 Cross reference
M 2 Cross reference
N 2 Cross reference
B 3 Cross reference
C 3 *** CHMK Change mode to kernel
D 3 *** CHMK Change mode to kernel
E 3 DECLARATIONS
F 3 CHANGE MODE TO KERNEL ROUTINE
G 3 CHANGE MODE TO KERNEL ROUTINE
H 3 ASTEXIT SYSTEM SERVICE
I 3 Local intermediate dispatches
J 3 Symbol table
K 3 Psect synopsis
L 3 Cross reference
M 3 Cross reference
N 3 Cross reference
B 4 Cross reference
C 4 DIAGNOSTIC SUPERVISOR COMMAND
D 4 DIAGNOSTIC SUPERVISOR COMMAND
E 4 DIAGNOSTIC SUPERVISOR COMMAND
F 4 DIAGNOSTIC SUPERVISOR COMMAND
G 4 DIAGNOSTIC SUPERVISOR COMMAND
H 4 DIAGNOSTIC SUPERVISOR COMMAND
I 4 DIAGNOSTIC SUPERVISOR COMMAND
J 4 DIAGNOSTIC SUPERVISOR COMMAND
K 4 DIAGNOSTIC SUPERVISOR COMMAND
L 4 DIAGNOSTIC SUPERVISOR COMMAND
M 4 DIAGNOSTIC SUPERVISOR COMMAND
N 4 DIAGNOSTIC SUPERVISOR COMMAND
B 5 Libraries, Macros, Equated Sym
C 5 Work Psect Storage
D 5 Data Psect Storage
E 5 Data Psect Storage
F 5 Commands A-C
G 5 Commands A-C
H 5 Directory, Deattach, Deselect,
I 5 Directory, Deattach, Deselect,
J 5 Commands Exit, Examine
K 5 Help, InitPCS, Load, Next, Run
L 5 Commands Set, Select, Show, St
M 5 End of Line Processing
N 5 End of Line Processing
B 6 End of Line Processing
C 6 Illegal Command Processing
D 6 Incomplete Command Processing
E 6 Bad List Processing
F 6 Bad Qualifier Processing
G 6 Get Generic Device Name
H 6 Deattach Processing
I 6 Deselect and Select Processing
J 6 Get a File Specification
K 6 Get a File Specification
L 6 Start and Run Processing
M 6 Start and Run Processing
N 6 Start and Run Processing
B 7 Start and Run Qualifiers
C 7 Start and Run Qualifiers
D 7 Show Parameters and Qualifiers
E 7 Show Parameters and Qualifiers
F 7 Show Parameters and Qualifiers
G 7 Show Parameters and Qualifiers
H 7 Show Parameters and Qualifiers
I 7 Show Parameters and Qualifiers
J 7 Show Parameters and Qualifiers
K 7 Show Parameters and Qualifiers
L 7 Show Parameters and Qualifiers
M 7 Clear Parameters and Qualifier
N 7 Clear Parameters and Qualifier
B 8 Clear Parameters and Qualifier
C 8 Clear Parameters and Qualifier
D 8 Clear Parameters and Qualifier
E 8 Clear Parameters and Qualifier
F 8 Set Parameters and Qualifiers
G 8 Set Parameters and Qualifiers
H 8 Set Parameters and Qualifiers
I 8 Set Parameters and Qualifiers
J 8 Set Parameters and Qualifiers
K 8 Set Parameters and Qualifiers
L 8 Set Parameters and Qualifiers
M 8 Set Parameters and Qualifiers
N 8 Set Parameters and Qualifiers
B 9 Set Parameters and Qualifiers
C 9 Examine Qualifiers and Paramet
D 9 Examine Qualifiers and Paramet
E 9 Examine Qualifiers and Paramet
F 9 Examine Qualifiers and Paramet
G 9 Examine Qualifiers and Paramet
H 9 Examine Qualifiers and Paramet
I 9 Deposit Parameters and Qualifi
J 9 Deposit Parameters and Qualifi
K 9 Deposit Parameters and Qualifi
L 9 Deposit Parameters and Qualifi
M 9 Deposit Parameters and Qualifi
N 9 Set Default Parameters
B 10 Set Default Parameters
C 10 Set Default Parameters
D 10 DS$Cli Routine - Command Line
E 10 DS$Cli Routine - Command Line
F 10 DS$Cli Routine - Command Line
G 10 DS$Cli Routine - Command Line
H 10 CLI Action Routines.
I 10 CLI Action Routines.
J 10 CLI Action Routines.
K 10 CLI Action Routines.
L 10 Null and Context-Saving Action
M 10 Null and Context-Saving Action
N 10 Success, Failure, Notnuf, Enuf
B 11 Commands A-C Action Routines
C 11 Commands D-E Action Routines
D 11 Commands D-E Action Routines
E 11 Commands H-R Action Routines
F 11 Commands H-R Action Routines
G 11 Commands S-Z Action Routines
H 11 CRD command
I 11 CRD command
J 11 Filespec, Optional, Required A
K 11 Start and Run Qualifier Action
L 11 Start and Run Qualifier Action
M 11 Event and Flags Action Routine
N 11 Base, Address, Break Action Ro
B 12 All, Default Action Routines
C 12 Show Calls Routine ; [70]
D 12 CRDTrace Action Routines ; [
E 12 Bell, Binary, Halt, IEx Action
F 12 Bell, Binary, Halt, IEx Action
G 12 Bell, Binary, Halt, IEx Action
H 12 Bell, Binary, Halt, IEx Action
I 12 Loop, Oper, Prompt Quick Actio
J 12 Search, Trace, Verify Action R
K 12 Set/Show Page, Width Action Ro
L 12 Set/Show QA Action Routines
M 12 QADef Action Routine
N 12 Efn, Next, Ascii Action Routin
B 13 Register Action Routines
C 13 Backup, Advance, Data Action R
D 13 Long, Word, Byte, Hex, Dec, Oc
E 13 IF Action Routines
F 13 IF Action Routines
G 13 IF Action Routines
H 13 Xdelta Action Routines
I 13 Device, Brief, Adaptor Action
J 13 Show Select, Do_Cmd, Set Load,
K 13 MM On, Off, and Show Action Ro
L 13 MM On, Off, and Show Action Ro
M 13 Show Status, Show Support
N 13 Show Status, Show Support
B 14 Symbol table
C 14 Symbol table
D 14 Symbol table
E 14 Symbol table
F 14 Symbol table
G 14 Symbol table
H 14 Psect synopsis
I 14 Cross reference
J 14 Cross reference
K 14 Cross reference
L 14 Cross reference

M 14 Cross reference
N 14 Cross reference
B 15 Cross reference
C 15 Cross reference
D 15 Cross reference
E 15 Cross reference
F 15 Cross reference
G 15 Cross reference
H 15 Cross reference
I 15 Cross reference
J 15 Cross reference
K 15 Cross reference
L 15 Cross reference
M 15 Cross reference
N 15 Cross reference
B 16 Cross reference
C 16 Cross reference
D 16 Cross reference
E 16 Cross reference
F 16 Cross reference
G 16 Cross reference
H 16 Cross reference
I 16 Cross reference
J 16 Cross reference
K 16 Cross reference
L 16 Cross reference

Table of contents

(1) 141 Macros and Equated Symbols
(1) 279 Work Psect Declarations
(1) 315 Data Psect Declarations
(1) 320 DS$CanWait - CANCEL DELAY ROUTINE
(1) 363 $WAKE - Wakeup from hibernate
(1) 397 DSX$WaitMS - MILLISECOND DELAY ROUTINE
(1) 478 DSX$WaitUC - MICROSECOND DELAY ROUTINE
(1) 570 EXE$HIBER - HIBERNATE SYSTEM SERVICE ROUTINE.
(1) 627 EXE$CANTIM - CANCEL TIMER SYSTEM SERVICE.
(1) 694 DSR$SETIMR_INI - SET TIMER QUEUE INITIALIZATION
(1) 772 EXE$Setimr - SET TIMER SYSTEM SERVICE.
(1) 862 DSI$TimSrv - INTERVAL TIMER INTERRUPT SERVICE ROUTINE.
(1) 923 DSI$TIMER - Level 7 timer interrupt service
(1) 1045 CLK$RE_INIT - SYSTEM TIMER INITIALIZATION SUBROUTINE

```
0000      1          .TITLE  CLOCK   INTERVAL TIMER ROUTINES.
0000      2          .IDENT  /07-30/
0000      3          .NoShow Conditionals
0000      4
0000      5 ;
0000      6 ; Copyright (c) 1977, 1982, 1983, 1984
0000      7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      8 ;
0000      9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     15 ; REMAIN IN DEC.
0000     16 ;
0000     17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     19 ; CORPORATION.
0000     20 ;
0000     21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     23
0000     24 ;++
0000     25 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     26 ;
0000     27 ; ABSTRACT:
0000     28 ;
0000     29 ; ENVIRONMENT:
0000     30 ;
0000     31 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000     32 ;
0000     33 ; MODIFIED BY:
0000     34 ;                KEN CHAPMAN     30-NOV-77       VERSION 02
0000     35 ;       01       FIXED $DS_WAITMS_L TO WAKE IN USER MODE.  (DSPR #65)
0000     36 ;                CHANGED $SETIMR TO RETURN A WARNING IF IPL IS TOO HIGH.
0000     37 ;                  (DSPR #53).
0000     38 ;
0000     39 ;                TOM SOUTTER     22-FEB-78       VERSION 03
0000     40 ;       02       INSTALLED EXE$GL_ABSTIM (NON-FUNCTIONAL)
0000     41 ;                TOM SOUTTER      2-MAR-78
0000     42 ;       03       CONVERTED SETIMR & TIMSRV ROUTINES TO USE VMS ASTDEL
0000     43 ;                TOM SOUTTER     23-MAR-78
0000     44 ;       04       TURN ON SYSTEM REAL TIME USING INTERVAL TIMER & TODC
0000     45 ;                ROGER RIGGS     03-APR-78
0000     46 ;       05       MODIFIED SETIMR TO QUE LONG REQUESTS SO THAT
0000     47 ;                THE CLOCK CAN BE USED FOR QIO WATCH DOG TIMERS
0000     48 ;       06       Fixed DSR$SETIMR_INI to release buffers in timer queue.
0000     49 ;                NICK HOWGATE    06-DEC-78       VERSION 4 (ESSAA-5.01)
0000     50 ;       07       EMULATE 'QUEUE' INSTRUCTIONS
0000     51 ;                COPY REQIDT TO ASTPRM IN SETIMR
0000     52 ;                AND FIX CANTIM TO BE ABLE TO CANCEL SPECIFIC TIMERS
0000     53 ;       08       Fix residual time in WAITMS to return correct value
0000     54 ;                Roger Riggs     11-Jun-1979
0000     55 ;       09       Revamped timer interrupt service to fix bug caused
0000     56 ;                by lowering IPL from 18 to 7 in calling SCH$QAST
0000     57 ;                when delivering timer requests. Also EXE$SETIMR
```

```
                        ;
                        ;               was affected, since the queue is now sorted by completion
                        ;               time
                        ;               Roger Riggs     20-NOV-1979
                        ;       10      Remove W^ from several references
                        ;               Roger Riggs, 24-Jan-1980, Version 5.2
                        ;       12      Removed Errsup after clock overflow in WAITUS, since
                        ;               it is unnecessary and means that we loaded the clock
                        ;               with a value that will overflow before we can check it.
                        ;               Also added DS$GK_USOVR, used by WAITUS and defined
                        ;               in ONLY7xx for each processor, Expermentially determined
                        ;               to produce an accuracy of about 1%.
                        ;               Also added clear of overflow bit to clock interrupt service
                        ;               Roger Riggs, 27-Feb-1980
                        ;       13      Changed DSBINT to SETIPL(#18) before removing entries from
                        ;               timer queue. Also restore IPL before continuing.
                        ;       14      Assured that Both SETIMR and CANTIM are executed in KERNEL mode
                        ;               Roger Riggs, 22-July-1980, Version 5.5
                        ;       15      Moved EXE$GL_ABSTIM to ENTRY.
                        ;
                        ;               Dave Butenhof, 16-oct-1980, version 6.1
                        ;       16      Move EXE$GQ_SYSTIME to entry, to make it accessible to drivers.
                        ;               Dave Butenhof, 18-feb-1981, version 6.3
                        ;       17      Delete SEP psect.
                        ;               Dave Butenhof, 25-feb-1981, version 6.3
                        ;       18      Fix problem found in queue handling.  REMQUE/INSQUE instructs
                        ;               had been emulated.  Restore the queue instructions.
                        ;       19      Fix WAITUS bug (Tom Kraus generated fix).
                        ;
                        ;       20      - Dave Butenhof, 26-Feb-1982, version 6.6
                        ;               Add special code in WAITUS for Nebula...use interrupt
                        ;               instead of polling.
                        ;
                        ;       21      - Jack Stansbury, 26-Feb-1982, Version 6.6
                        ;               Added .LIBRARY statements for $DS, $DIAG, and $LIB.
                        ;
                        ;       22      Jack Stansbury, 15-September-1982, Version 6.9
                        ;               Changed ICCS$M_ERR from 31 to 15. Also added code in DSI$TimSrv
                        ;               routine that will not reset the interval timer if the SetVec routine
                        ;               is currently trying to stop the interval timer.
                        ;
                        ;       23      Jack Stansbury, 16-Septe mber-1982, Version 6 9
                        ;               Changed the ICCS$M_ERR bit back to #15. The VAX Hardware Handbook
                        ;               is wrong in its picture of the ICCS in the back of the book.
                        ;
                        ;       24      Marion Baggett, 24-Sept-1982,   Version 6.9
                        ;               Changed the entry mask in DSX$WAITUS to not save R2, R3. These regis
                        ;               are not changed, and the extra time for CAllx to save them should be
                        ;       25      Bob Bergazzi    21-Oct-1982     Version 6.9
                        ;               Changed DSJ$TIMSRV to clear the interrupt when the
                        ;               DS$GB_Stopping_The_Timer flag is set. Otherwise we get stuck
                        ;               in a loop of continuous interrupts.
                        ;
                        ;       26      Bob Bergazzi    25-April-1983   Version 6.11
                        ;               Added global symbol DSX$WAITUS_USOVR to the instruction in the
                        ;               DSX$WAITUS routine which computes the interval clock count,
                        ;               so that we can reference it from KERNEL and change the value
                        ;               of the clock routine overhead factor (DS$GK_USOVR).
```

```
0000    115 ;               if we are running on a SUPERSTAR processor. This is done from
0000    116 ;               KERNEL during the BOOT procedure so that we don't add any more
0000    117 ;               instructions to the DSX$WAITUS routine, which would change the
0000    118 ;               value of the overhead factor for the existing processors.
0000    119 ;               I.e. The code in KERNEL is modifying the instruction at
0000    120 ;               DSX$WAITUS_USOVR.
0000    121 ;
0000    122 ;       27      Bob Bergazzi    May 16,1983     Version 6.11
0000    123 ;               Changed the order of the .LIB statements.
0000    124 ;
0000    125 ;       28      Richard Brown   August 4, 1983  Version 6.13
0000    126 ;               Corrected bug in DSX$CANWAIT.  If this service was called
0000    127 ;               to cancel a $DS_WAITMS, it did not remove the entry from the
0000    128 ;               timer queue, so the AST was still being delivered.  Therefore,
C000    129 ;               the $SETIMR call in DSX$WAITMS was changed so that a request
0000    130 ;               ID is specified (-1).  DSX$CANWAIT was changed so that it
0000    131 ;               issues a $CANTIM to kill all timer requests with an ID of -1.
0000    132 ;
0000    133 ;       29      R.E.Muse        10 January 1984 version 6.14
0000    134 ;               added PRDEF macro definition - Version 4 changes.
0000    135 ;
0000    136 ;       30      Bob Bergazzi    Jan. 26, 1984   Version 6.14
0000    137 ;               Changed CLK$RE_INIT routine to call ONLY_CLOCK_INIT which
0000    138 ;               does cpu-specific stuff to the TODR.
0000    139 ;--
```

```
0000    141          .SBTTL  Macros and Equated Symbols
0000    142 ;
0000    143 ; INCLUDE FILES:
0000    144 ;
0000    145
0000    146          .LIBRARY        /SYS$LIBRARY:LIB/       ;                       [27]
0000    147          .LIBRARY        /$DS/                   ;                       [27]
0000    148          .LIBRARY        /$DIAG/                 ;                       [27]
0000    149 ;
0000    150 ; MACROS
0000    151 ;
0000    152          .MACRO  $PRDEF,$GBL
0000    153
0000    154          $DEFINI PR,$GBL
0000    155
0000    156
0000    157  $EQU    PR$_KSP 0
0000    158  $EQU    PR$_ESP 1
0000    159  $EQU    PR$_SSP 2
0000    160  $EQU    PR$_USP 3
0000    161  $EQU    PR$_ISP 4
0000    162  $EQU    PR$_P0BR        8
0000    163  $EQU    PR$_P0LR        9
0000    164  $EQU    PR$_P1BR        10
0000    165  $EQU    PR$_P1LR        11
0000    166  $EQU    PR$_SBR 12
0000    167  $EQU    PR$_SLR 13
0000    168  $EQU    PR$_PCBB        16
0000    169  $EQU    PR$_SCBB        17
0000    170  $EQU    PR$_IPL 18
0000    171  $EQU    PR$_ASTLVL      19
0000    172  $EQU    PR$_SIRR        20
0000    173  $EQU    PR$_SISR        21
0000    174  $EQU    PR$_ICCS        24
0000    175  $EQU    PR$_NICR        25
0000    176  $EQU    PR$_ICR 26
0000    177  $EQU    PR$_TODR        27
0000    178  $EQU    PR$_RXCS        32
0000    179  $EQU    PR$_RXDB        33
0000    180  $EQU    PR$_TXCS        34
0000    181  $EQU    PR$_TXDB        35
0000    182  $EQU    PR$_ACCS        40
0000    183  $EQU    PR$_ACCR        41
0000    184  $EQU    PR$_MAPEN       56
0000    185  $EQU    PR$_TBIA        57
0000    186  $EQU    PR$_TBIS        58
0000    187  $EQU    PR$_PME 61
0000    188  $EQU    PR$_SID 62
0000    189  $EQU    PR$_TBCHK       63
0000    190  $EQU    PR$V_SID_SN     0
0000    191  $EQU    PR$S_SID_SN     12
0000    192  $EQU    PR$V_SID_PL     12
0000    193  $EQU    PR$S_SID_PL     3
0000    194  $EQU    PR$V_SID_ECO    15
0000    195  $EQU    PR$S_SID_ECO    9
0000    196  $EQU    PR$V_SID_TYPE   24
0000    197  $EQU    PR$S_SID_TYPE   8
```

```
0000    198  $EQU    PR$_SID_TYP780  1
0000    199  $EQU    PR$_SID_TYP750  2
0000    200  $EQU    PR$_SID_TYP730  3
0000    201  $EQU    PR$_SID_TYP7VV  4
0000    202  $EQU    PR$_SID_TYPMAX  4
0000    203  $EQU    PR$_WCSA        44
0000    204  $EQU    PR$_WCSD        45
0000    205  $EQU    PR$_SBIFS       48
0000    206  $EQU    PR$_SBIS        49
0000    207  $EQU    PR$_SBISC       50
0000    208  $EQU    PR$_SBIMT       51
0000    209  $EQU    PR$_SBIER       52
0000    210  $EQU    PR$_SBITA       53
0000    211  $EQU    PR$_SBIQC       54
0000    212  $EQU    PR$_CMIERR      23
0000    213  $EQU    PR$_CSRS        28
0000    214  $EQU    PR$_CSRD        29
0000    215  $EQU    PR$_CSTS        30
0000    216  $EQU    PR$_CSTD        31
0000    217  $EQU    PR$_TBDR        36
0000    218  $EQU    PR$_CADR        37
0000    219  $EQU    PR$_MCESR       38
0000    220  $EQU    PR$_CAER        39
0000    221  $EQU    PR$_UBRESET     55
0000    222  $EQU    PR$_PAMACC      64
0000    223  $EQU    PR$_PAMLOC      65
0000    224  $EQU    PR$_CSWP        66
0000    225  $EQU    PR$_CRBT        67
0000    226  $EQU    PR$_MCTL1       68
0000    227  $EQU    PR$_MCTL2       69
0000    228  $EQU    PR$_MGEN        70
0000    229  $EQU    PR$_MTBER       71
0000    230  $EQU    PR$_MEAR        72
0000    231  $EQU    PR$_MDCR1       73
0000    232  $EQU    PR$_MEDR        74
0000    233  $EQU    PR$_MECCR       75
0000    234
0000    235          $DEFEND PR,$GBL,DEF
0000    236
0000    237          .ENDM   $PRDEF
0000    238
0000    239  ;
0000    240  ; EQUATED SYMBOLS:
0000    241  ;
0000    242
0000    243          $CANTIMDEF
0000    244          $GETIMDEF
0000    245          $DDBDEF
0000    246          $DS_DSADEF
0000    247          $DS_DSDEF
0000    248          $DYNDEF
0000    249          $IPLDEF
0000    250          $PRDEF
0000    251          $DS_SCBDEF
0000    252          $SSDEF
0000    253          $PSLDEF
0000    254          $SETIMRDEF
```

```
                     0000   255          $TQEDEF
                     0000   256          $UCBDEF
                     0000   257          $DS_WAITMS_DEF
                     0000   258          $DS_WAITUS_DEF
                     0000   259          CMKDEF
                     0000   260          DSFDEF
                     0000   261
00000001             0000   262 ICCS$M_RUN      = 1 @ 0
00000010             0000   263 ICCS$M_XFER     = 1 @ 4
00000040             0000   264 ICCS$M_IE       = 1 @ 6
00000080             0000   265 ICCS$M_INT      = 1 @ 7
80000000             0000   266 ICCS$M_ERR      = 1 @ 31                    ;                                    [23]
                     0000   267
800000C1             0000   268 ICCS$M_RESET    = <ICCS$M_ERR ! ICCS$M_INT ! ICCS$M_IE ! ICCS$M_RUN>
                     0000   269
10000000             0000   270 TODR_BIAS       = 1 @ 28
                     0000   271
00002710             0000   272 TICK_US         = 10000                     ; MICROSECONDS PER TICK
000186A0             0000   273 TICK_NS         = TICK_US*10                ; 100NS UNITS PER TICK
00000064             0000   274 SEC_TICK       == 1000000/TICK_US           ; TICKS PER SECOND
                     0000   275
00000018             0000   276 CLOCK_IPL       = 24
```

```
                                00000000   278          .PSECT  Work, NoShr, NoExe, Wrt, Long   ;                                  [17]
                                    0000   279          .SBTTL  Work Psect Declarations
                                    0000   280 ;
                                    0000   281 ; OWN STORAGE:
                                    0000   282 ;
                                    0000   283
                                    0000   284 DS$GL_TIMQFL:
                         00000000'  0000   285          .LONG   DS$GL_TIMQFL            ;QUEUE HEADER FOR LONG SETIMR REQUESTS
                         00000000'  0004   286          .LONG   DS$GL_TIMQFL
                                    0008   287
                                    0008   288 DS$GQ_CURYEAR::
                         00000010   0008   289          .BLKQ   1
                                    0010   290
                                    0010   291 DS$GT_NEWYEAR::
20 38 37 39 31 2D 4E 41 4A 2D 31 30 0010   292          .ASCII  \01-JAN-1978  0:0:0.0\
            30 2E 30 3A 30 3A 30 20 001C
                                    0024   293
                         00000014   0024   294 DS$K_NYSIZ  ==  .-DS$GT_NEWYEAR
                                    0024   295
                                    0024   296 ONESECTIM:
                00000000 00000000   0024   297          .LONG   0,0                     ; Flink and Blink
                            0000    002C   298          .WORD   0                       ; Length =0
                              0F    002E   299          .BYTE   DYN$C_TQE               ; Type of data struncture
                              05    002F   300          .BYTE   TQE$C_SSREPT            ; Request type=repeat
                         00000380'  0030   301          .LONG   ONE_SECOND              ; Address of once/sec routine
                00000000 00000000   0034   302          .LONG   0,0                     ; Unused longwords
                00000000 00000000   003C   303          .LONG   0,0                     ; Initial expiration time
                        00989680    0044   304          .LONG   100000*100              ; Delta time=1 second
                        00000000    0048   305          .LONG   0                       ; Final unused
                                    004C   306
                                    004C   307 WAITMS_ARGS:
                                    004C   308          $SETIMR  DAYTIM=MSTIME, ADRADR=DS$CANWAIT, -
                                    004C   309                   REQIDT=-1                                          ;[28]
                                    0060   310
                                    0060   311 MSTIME:
                         00000068   0060   312          .BLKL   2
                                    0068   313
```

```
                     0068   315          .SBTTL  Data Psect Declarations
                00000000    316          .PSECT  Data, Shr, NoExe, NoWrt, Byte
                     0000   317
                     0000   318          MODNAM  CLOCK
```

```
                     0006   320          .SBTTL  DS$CanWait -    CANCEL DELAY ROUTINE
                 00000000   321          .PSECT  Code, Shr, Exe, NoWrt, Long     ;                                  [17]
                     0000   322 ;++
                     0000   323 ; FUNCTIONAL DESCRIPTION:
                     0000   324 ;
                     0000   325 ;       ROUTINE TO CANCEL PROGRAM DELAY
                     0000   326 ;
                     0000   327 ; CALLING SEQUENCE:
                     0000   328 ;
                     0000   329 ;       DS$CANWAIT()
                     0000   330 ;
                     0000   331 ; INPUT PARAMETERS:     NONE
                     0000   332 ;
                     0000   333 ; IMPLICIT INPUTS:      NONE
                     0000   334 ;
                     0000   335 ; OUTPUT PARAMETERS:    NONE
                     0000   336 ;
                     0000   337 ; IMPLICIT OUTPUTS:
                     0000   338 ;
                     0000   339 ;       DS$GL_FLAGS:
                     0000   340 ;               DS$V_ABRTFLG    = ABORT WAIT FLAG.
                     0000   341 ;
                     0000   342 ; COMPLETION CODES:
                     0000   343 ;
                     0000   344 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                     0000   345 ;
                     0000   346 ; SIDE EFFECTS:
                     0000   347 ;
                     0000   348 ;       WAKES UP HIBERNATING PROCESS.
                     0000   349 ;--
                     0000   350
                0000 0000   351 .ENTRY  DSX$CANWAIT,^M<>
                     0002   352
                     0002   353          $CANTIM_S       REQIDT=#-1      ; Dequeue the timer queue entry  [28]
                     0011   354                                          ; that was enqueued by the       [28]
                     0011   355                                          ; $SETIMR call in DSX$WAITMS     [28]
                     0011   356                                          ; (If we're not cancelling a     [28]
                     0011   357                                          ; $DS_WAITMS, this doesn't do    [28]
                     0011   358                                          ; anything.)                     [28]
             7E      7C 0011   359          CLRQ    -(SP)               ; Two null args
 00000000'EF         02 0013   360          CALLS   #2,SYS$WAKE         ; wake the program
                     04 001A   361          RET                         ; RETURN TO THE DIAGNOSTIC
```

```
                     001B   363          .SBTTL  $WAKE - Wakeup from hibernate
                     001B   364 ;++
                     001B   365 ; FUNCTIONAL DESCRIPTION:
                     001B   366 ;
                     001B   367 ;       This routine releases the process blocked by $HIBER
                     001B   368 ;
                     001B   369 ; CALLING SEQUENCE:
                     001B   370 ;
                     001B   371 ;       SYS$WAKE()
                     001B   372 ;
                     001B   373 ; INPUT PARAMETERS:     NONE
                     001B   374 ;
                     001B   375 ; IMPLICIT INPUTS:      NONE
                     001B   376 ;
                     001B   377 ; OUTPUT PARAMETERS:    NONE
                     001B   378 ;
                     001B   379 ; IMPLICIT OUTPUTS:
                     001B   380 ;
                     001B   381 ;       DS$V_ABRTFLG is SET in DS$GL_FLAGS
                     001B   382 ;
                     001B   383 ; COMPLETION CODES:     SS$_NORMAL
                     001B   384 ;
                     001B   385 ; SIDE EFFECTS:
                     001B   386 ;
                     001B   387 ;       The next time or current time $HIBER is called it will exit.
                     001B   388 ;--
                     001B   389
                0000 001B    390 .ENTRY  EXE$WAKE,^M<>
                     001D   391
0000'CF    40 8F     88 001D   392          BISB    #DS$M_ABRTFLG, -
                     0023   393                  W^DS$GL_FLAGS           ; Set the flag
          50   01    D0 0023   394          MOVL    S^#SS$_NORMAL,R0        ; It worked
                     04 0026   395          RET                             ; Return
```

```
                     0027   397          .SBTTL  DSX$WaitMS -    MILLISECOND DELAY ROUTINE
                     0027   398 ;++
                     0027   399 ; FUNCTIONAL DESCRIPTION:
                     0027   400 ;
                     0027   401 ;       THIS ROUTINE ALLOWS THE PROGRAMMER TO DELAY HIS PROGRAM SOME
                     0027   402 ;       NUMBER OF 10 MILLISECOND UNITS.
                     0027   403 ;
                     0027   404 ; CALLING SEQUENCE:
                     0027   405 ;
                     0027   406 ;       DIAGNOSTIC SUPERVISOR PROCEDURE CALL.
                     0027   407 ;
                     0027   408 ; INPUT PARAMETERS:
                     0027   409 ;
                     0027   410 ;       WAITMS$_TIME(AP)= NUMBER OF 10 MILLISECOND UNITS TO DELAY.
                     0027   411 ;       WAITMS$_TAG(AP) = ADDRESS TO RECEIVE REMAINING TIME ON CANWAIT.
                     0027   412 ;
                     0027   413 ; IMPLICIT INPUTS:
                     0027   414 ;
                     0027   415 ;       NONE
                     0027   416 ;
                     0027   417 ; OUTPUT PARAMETERS:
                     0027   418 ;
                     0027   419 ;       R0 = SUCCESS/FAILURE
                     0027   420 ;       R1 = NUMBER OF UNITS LEFT
                     0027   421 ;
                     0027   422 ; IMPLICIT OUTPUTS:
                     0027   423 ;
                     0027   424 ;       NONE
                     0027   425 ;
                     0027   426 ; COMPLETION CODES:
                     0027   427 ;
                     0027   428 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                     0027   429 ;       DS$_ICERR       = INTERVAL CLOCK ERROR.
                     0027   430 ;       DS$_PROGERR     = NEGATIVE TIME INTERVAL SPECIFIED.
                     0027   431 ;                           (OR LESS THAN OVERHEAD)
                     0027   432 ;
                     0027   433 ; SIDE EFFECTS:
                     0027   434 ;
                     0027   435 ;       NONE
                     0027   436 ;--
```

```
                                0000  0027   438 .ENTRY  DSX$WAITMS,^M<>
                                      0029   439
                                      0029   440         $GETTIM_S  -(SP)                    GET START TIME
                                      0032   441
                           50    DC   0032   442         MOVPSL  R0                          ; READ PROCESSOR STATUS.
                     05    10    ED   0034   443         CMPZV   #PSL$V_IPL, #PSL$S_IPL,-    ; CHECK IPL.
                     02    50         0037   444                   R0, #2
                           19    19   0039   445         BLSS    10$
     50    000003E8 8F    04 AC    C5   003B   446         MULL3   WAITMS$_TIME(AP), -         ; CHANGE MILLSECONDS TO MICROSECONDS.
                                      0044   447                   #1000, R0
                                      0044   448         $DS_WAITUS_S  TIME=R0
                        5D 50    E9   004F   449         BLBC    R0,DSX$WAITMS_X             ; ERROR IN WAITUS, EXIT
                           2E    11   0052   450         BRB     30$
                                      0054   451
           50    00060020 8F    D0   0054   452 10$:     MOVL    #DS$_PROGERR, R0            ; ANTICIPATE NEG. TIME
        FFFE7960 8F    04 AC    7A   005B   453         EMUL    WAITMS$_TIME(AP), #-TICK_NS,- ; TRANSLATE TIME INTO 100NS INCS.
00000060'EF    00001200 8F         0063   454                   #4608, MSTIME               ;  PLUS OVERHEAD
                           40    18   006D   455         BGEQ    DSX$WAITMS_X                ; IF GEQ, NEGATIVE OR TOO SMALL
                                      006F   456
                                      006F   457         $SETIMR_G  W^WAITMS_ARGS
                        34 50    E9   0078   458         BLBC    R0, DSX$WAITMS_X            ; SKIP IF ERROR.
                                      007B   459
                                      007B   460         $HIBER_S
                                      0082   461
                                      0082   462 30$:     $GETTIM_S  -(SP)                    ; GET END TIME
                     50    8E    7D   008B   463         MOVQ    (SP)+,R0                    ; GET END TIME OFF STACK
                     50    8E    C2   008E   464         SUBL2   (SP)+,R0                    ; MAKE LOW ORDER DIFFERENCE
                     51    8E    D9   0091   465         SBWC    (SP)+,R1                    ; AND HIGH ORDER DIFFERENCE
50   51    50    000186A0 8F    7B   0094   466         EDIV    #^D100000,R0,R1,R0          ; MAKE TRASH(R1) AND 10'S OF MILLSECONDS
                      50    08 AC    D0   009D   467         MOVL    WAITMS$_TAG(AP),R0          ; CHECK FOR REMAINING TIME ADR.
                           09    13   00A1   468         BEQL    40$
                60    04 AC       51   C3   00A3   469         SUBL3   R1,WAITMS$_TIME(AP),(R0); Store unused time
                                 02   18   00A8   470         BGEQ    40$                         ; Branch if positive
                                 60   D4   00AA   471         CLRL    (R0)                        ; Was negative reduce to zero
                                      00AC   472
                           50    01   D0   00AC   473 40$:     MOVL    S^#SS$_NORMAL,R0            ; NORMAL RETURN
                                      00AF   474
                                      00AF   475 DSX$WAITMS_X:
                                 04   00AF   476         RET                                 ; RETURN TO PROGRAM
```

```
                     00B0    478          .SBTTL  DSX$WaitUC -    MICROSECOND DELAY ROUTINE
                     00B0    479 ;++
                     00B0    480 ; FUNCTIONAL DESCRIPTION:
                     00B0    481 ;
                     00B0    482 ;       THIS ROUTINE ALLOWS THE PROGRAMMER TO DELAY HIS PROGRAM SOME
                     00B0    483 ;       NUMBER OF 10 MICROSECOND UNITS.
                     00B0    484 ;
                     00B0    485 ; CALLING SEQUENCE:
                     00B0    486 ;
                     00B0    487 ;       DIAGNOSTIC SUPERVISOR PROCEDURE CALL.
                     00B0    488 ;
                     00B0    489 ; INPUT PARAMETERS:
                     00B0    490 ;
                     00B0    491 ;       WAITUS$_TIME(AP)= NUMBER OF 10 MICROSECOND UNITS TO DELAY.
                     C0B0    492 ;       WAITUS$_TAG(AP) = ADDRESS TO RECEIVE REMAINING TIME ON CANWAIT.
                     00B0    493 ;
                     00B0    494 ; IMPLICIT INPUTS:
                     00B0    495 ;
                     00B0    496 ;       NONE
                     00B0    497 ;
                     00B0    498 ; OUTPUT PARAMETERS:
                     00B0    499 ;
                     00B0    500 ;       R0 = SUCCESS/FAILURE
                     00B0    501 ;       R1 = NUMBER OF UNITS LEFT
                     00B0    502 ;
                     00B0    503 ; IMPLICIT OUTPUTS:
                     00B0    504 ;
                     00B0    505 ;       NONE
                     00B0    506 ;
                     00B0    507 ; COMPLETION CODES:
                     00B0    508 ;
                     00B0    509 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                     00B0    510 ;       DS$_ICERR       = INTERVAL CLOCK ERROR.
                     00B0    511 ;       DS$_PROGERR     = NEGATIVE TIME INTERVAL SPECIFIED.
                     00B0    512 ;                           (OR LESS THAN OVERHEAD)
                     00B0    513 ;
                     00B0    514 ; SIDE EFFECTS:
                     00B0    515 ;
                     00B0    516 ;       NONE
                     00B0    517 ;--
```

```
                                00B0   519          .ENABLE LOCAL_BLOCK                ;                                        [26]
                           0000 00B0   520 .ENTRY   DSX$WAITUS,^M<>                    ;                                        [24]
                                00B2   521          DsbInt  #31                        ; Set to IPL 31                          [20]
                     50   1C   D0 00B8   522          MOVL    S^#SS$_EXQUOTA, R0         ; ASSUME MULTIPLE TIME FUNCTION ERROR
          00000000'EF   11   E0 00BB   523          BBS     #DS$V_TIMRON, -            ; CHECK FOR TIMER ALREADY IN USE.
                        1F      00C2   524                  DS$GL_FLAGS,25$            ; LEAP FROG TO DSX$WAITUS_X
                                00C3   525
          00000000'EF   40 8F   8A 00C3   526          BICB2   #DS$M_ABRTFLG, -           ; CLEAR ABORT WAIT FLAG.
                                00CB   527                  DS$GL_FLAGS
                                00CB   528 DSX$WAITUS_USOVR::                           ;                                        [26]
          FFFFFFF6 8F   04 AC   7A 00CB   529          EMUL    WAITUS$_TIME(AP), -        ; COMPUTE INTERVAL CLOCK COUNT.
             50   00000000'8F      00D3
                                00D9   530                  #-10,#DS$GK_USOVR,R0       ; PLUS OVERHEAD
                           0C   19 00D9   531          BLSS    30$
             50   00660020 8F   D0 00DB   532          MOVL    #DS$_PROGERR, R0           ; IF GEQ, NEGATIVE OR TOO SMALL
                                00E2   533 25$:      Enbint                             ; Restore IPL on error path              [20]
                           49   11 00E5   534          BRB     DSX$WAITUS_X
                                00E7   535
             18   80000080 8F   DA 00E7   536 30$:      MTPR    #ICCS$M_INT + -            ; STOP THE CLOCK.
                                00EE   537                   ICCS$M_ERR, #PR$_ICCS
                           51   D4 00EE   538          CLRL    R1                         ; ANTICIPATE NO RESIDUAL TIME.
                        19   50   DA 00F0   539          MTPR    R0, #PR$_NICR              ; SET NEXT INTERVAL.
             18   80000091 8F   DA 00F3   540          MTPR    #ICCS$M_XFER ! ICCS$M_RUN ! -   ;                           [19]
                                00FA   541                   ICCS$M_INT ! ICCS$M_ERR, -     ;                           [19]
                                00FA   542                     #PR$_ICCS                ; LOAD AND START THE INTERVAL CLOCK.
                                00FA   543          Enbint                             ; Enable interrupts again                [20]
                                00FD   544
          00000000'EF   06   E4 00FD   545 40$:      BBSC    #DS$V_ABRTFLG, -           ; CHECK FOR CANWAIT.
                           0E      0104   546                  DS$GL_FLAGS, 50$
                     50   18   DB 0105   547          MFPR    #PR$_ICCS, R0              ; READ THE CLOCK STATUS.
          50   80000080 8F   D3 0108   548          BITL    #ICCS$M_INT ! -            ; CHECK FOR DONE
                                010F   549                   ICCS$M_ERR, R0            ;  OR ERROR.
                           EC   13 010F   550          BEQL    40$                        ; BR IF NEITHER.
                           0A   11 0111   551          BRB     55$                        ; Normal completion, no residual         [20]
                                0113   552
                     51   1A   DB 0113   553 50$:      MFPR    #PR$_ICR, R1               ; SAVE COUNT REMAINING.
          51   FFFFFFF6 8F   C6 0116   554          DIVL    #-10, R1                   ; COMPUTE 10US UNITS LEFT.
                                011D   555
                        08 AC   D5 011D   556 55$:      TSTL    WAITUS$_TAG(AP)            ; IF RETURN TAG NOT SPECIFIED            [20]
                           04   13 0120   557          BEQL    60$                        ; SKIP THE STORE
                     08 BC   51   D0 0122   558          MOVL    R1,@WAITUS$_TAG(AP)        ; STORE THE REMAINING (10US)
                                0126   559
                     50   01   D0 0126   560 60$:      MOVL    S^#SS$_NORMAL, R0                  ; NORMAL RETURN.
                                0129   561
                        03   BB 0129   562 70$:      PUSHR   #^M<R0,R1>                 ; SAVE RETURN INFORMATION
                     02AA   30 012B   563          BSBW    CLK$RE_INIT                ; RESTART AND RESYNC THE TOD AND ICCS
                        03   BA 012E   564          POPR    #^M<R0,R1>                 ; RESTORE RETURN INFORMATION
                                0130   565          .DISABLE LOCAL_BLOCK               ;                                        [26]
                                0130   566
                                0130   567 DSX$WAITUS_X:
                           04   0130   568          RET                                ; RETURN TO PROGRAM
```

```
                     570          .SBTTL  EXE$HIBER -     HIBERNATE SYSTEM SERVICE ROUTINE.
0131    571 ;++
0131    572 ; FUNCTIONAL DESCRIPTION:
0131    573 ;
0131    574 ;       EMULATION OF STARLET SYSTEM SERVICE ''HIBER'' (HIBERNATE).
0131    575 ;       THE HIBERNATE SYSTEM SERVICE ALLOWS A PROCESS TO MAKE ITSELF INACTIVE
0131    576 ;       BUT TO REMAIN KNOWN TO THE SYSTEM SO THAT IT CAN BE INTERRUPTED, FOR
0131    577 ;       EXAMPLE TO RECEIVE ASTS.  A HIBERNATE REQUEST IS A WAIT-FOR-WAKE-EVENT
0131    578 ;       REQUEST.  WHEN A WAKE IS ISSUED FOR A HIBERNATING PROCESS WITH THE
0131    579 ;       $WAKE SYSTEM SERVICE OR A RESULT OF A SCHEDULE WAKEUP ($SCHDWK)
0131    580 ;       SYSTEM SERVICE, THE PROCESS CONTINUES EXECUTION AT THE INSTRUCTION
0131    581 ;       FOLLOWING THE HIBERNATE CALL.
0131    582 ;
0131    583 ; CALLING SEQUENCE:
C131    584 ;
0131    585 ;       STARLET PROCEDURE CALL.
0131    586 ;
0131    587 ; INPUT PARAMETERS:     NONE
0131    588 ;
0131    589 ; IMPLICIT INPUTS:      NONE
0131    590 ;
0131    591 ; OUTPUT PARAMETERS:    NONE
0131    592 ;
0131    593 ; IMPLICIT OUTPUTS:     NONE
0131    594 ;
0131    595 ; COMPLETION CODES:
0131    596 ;
0131    597 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
0131    598 ;
0131    599 ; SIDE EFFECTS:         NONE
0131    600 ;--
```

```
                      0000  0131   602 .ENTRY  EXE$HIBER,^M<>
                            0133   603
                  FECA'  30 0133   604 10$:    BSBW    KB_CHECK                ; CHECK FOR ^C.
00000000'EF         06  E5 0136   605         BBCC    #DS$V_ABRTFLG, -        ; CHECK FOR ABORTWAIT FLAG.
                    F5       013D   606                 DS$GL_FLAGS, 10$
                             013E   607         CMK     HIBER                   ; CLEAR CLOCK.
               50   01  D0 014D   608         MOVL    S^#SS$_NORMAL, R0       ; NORMAL RETURN STATUS.
                            0150   609 EXE$HIBER_X:
                        04  0150   610         RET
                            0151   611
                            0151   612 ;
                            0151   613 ; KERNEL MODE HIBERNATE ROUTINE.
                            0151   614 ;
                            0151   615
                            C151   616 CMK$HIBER::
00000000'EF         11  E5 0151   617         BBCC    #DS$V_TIMRON, -         ; RELEASE THE TIMER.
                    0D       0158   618                 DS$GL_FLAGS, CMK$HIBER_X
18   80000080  8F   DA 0159   619         MTPR    #ICCS$M_ERR + -         ; IF TIMED, STOP & CLEAR CLOCK.
                            0160   620                  ICCS$M_INT, #PR$_ICCS
               51   1A  DB 0160   621         MFPR    #PR$_ICR, R1            ; SAVE COUNT.
                  0272  30 0163   622         BSBW    CLK$RE_INIT
                            0166   623
                            0166   624 CMK$HIBER_X:
                        02  0166   625         REI                             ; RETURN.
```

```
                     0167   627          .SBTTL  EXE$CANTIM -    CANCEL TIMER SYSTEM SERVICE.
                     0167   628 ;++
                     0167   629 ; FUNCTIONAL DESCRIPTION:
                     0167   630 ;
                     0167   631 ;       THE CANCEL TIMER REQUEST SYSTEM SERVICE CANCELS ALL OR A SELECTED
                     0167   632 ;       SUBSET OF THE SET TIMER REQUESTS PREVIOUSLY ISSUED BY A PROCESS.
                     0167   633 ;
                     0167   634 ; CALLING SEQUENCE:
                     0167   635 ;
                     0167   636 ;       STARLET PROCEDURE CALL.
                     0167   637 ;
                     0167   638 ; INPUT PARAMETERS:
                     0167   639 ;
                     0167   640 ;       REQIDT  = If zero cancel all, else match with ASTPRM.
                     0167   641 ;       ACMODE  = NOT IMPLEMENTED.
                     0167   642 ;
                     0167   643 ; IMPLICIT INPUTS:       NONE
                     0167   644 ;
                     0167   645 ; OUTPUT PARAMETERS:     NONE
                     0167   646 ;
                     0167   647 ; IMPLICIT OUTPUTS:      NONE
                     0167   648 ;
                     0167   649 ; COMPLETION CODES:
                     0167   650 ;
                     0167   651 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                     0167   652 ;
                     0167   653 ; SIDE EFFECTS:          NONE
                     0167   654 ;--
```

```
                          003C  0167    656 .ENTRY  EXE$CANTIM,^M<R2,R3,R4,R5>
                                0169    657
                                0169    658         CMK     CANTIM                  ; Change to KERNEL mode
                    50    01    D0      0178    659         MOVL    S^#SS$_NORMAL,R0        ; Success
                                04      017B    660         RET
                                        017C    661
                                        017C    662 ;+
                                        017C    663 ; KERNEL mode routine to cancel timer queues
                                        017C    664 ;-
                                        017C    665 CMK$CANTIM::
                                        017C    666         DSBINT                  ; Prevent interruptions
               55   00000000'EF    DE      0182    667         MOVAL   DS$GL_TIMQFL,R5         ; Point to queue head
                     54    04 AC   D0      0189    668         MOVL    CANTIM$_REQIDT(AP),R4   ; Get request identification
                                        018D    669
                           65    DD      018D    670         PUSHL   (R5)                    ; Push address of first FLINK
                                        018F    671
                           50 8ED0       018F    672 10$:    POPL    R0                      ; Get address of entry
                     55    50    D1      0192    673         CMPL    R0,R5                   ; FLINK to queue head?
                           19    13      0195    674         BEQL    30$                     ; All done
                           60    DD      0197    675         PUSHL   TQE$L_TQFL(R0)          ; Push link to next TQE
                                        0199    676
                        0B A0    95      0199    677         TSTB    TQE$B_RQTYPE(R0)        ; Is it ast request?
                           F1    12      019C    678         BNEQ    10$                     ; No, leave it in the queue
                           54    D5      019E    679         TSTL    R4                      ; Cancel all?
                           06    13      01A0    680         BEQL    20$                     ; Branch if all
                     14 A0    54    D1      01A2    681         CMPL    R4,TQE$L_ASTPRM(R0)     ; Match on REQIDT?
                           E7    12      01A6    682         BNEQ    10$                     ; Branch if not
                                        01A8    683 ;+
                                        01A8    684 ; REMOVE ENTRY IN R0 FROM QUEUE AND DEALLOCATE
                                        01A8    685 ;-
               50    60    0F      01A8    686 20$:    REMQUE  (R0), R0                ; Remove queue entry                    [18]
                  FE52'    30      01AB    687         BSBW    EXE$DEANONPAGED         ; DE-ALLOCATE THE SPACE
                     DF    11      01AE    688         BRB     10$                     ; Check next
                                        01B0    689
                                        01B0    690 30$:
                                        01B0    691         ENBINT                  ; Restore IPL
                           02   01B3    692         REI                             ; Return to previous mode
```

```
                        01B4    694          .SBTTL  DSR$SETIMR_INI -        SET TIMER QUEUE INITIALIZATION
                        01B4    695 ;++
                        01B4    696 ; FUNCTIONAL DESCRIPTION:
                        01B4    697 ;
                        01B4    698 ;       THIS ROUTINE INITIALIZES THE SETIMR QUEUES
                        01B4    699 ;       AND IS CALLED FROM KERNEL ON A START 10000
                        01B4    700 ;
                        01B4    701 ; CALLING SEQUENCE:
                        01B4    702 ;
                        01B4    703 ;       BSBW    DSR$SETIMR_INI
                        01B4    704 ;
                        01B4    705 ; INPUT PARAMETERS:     NONE
                        01B4    706 ;
                        01B4    707 ; IMPLICIT INPUTS:      NONE
                        01B4    708 ;
                        01B4    709 ; OUTPUT PARAMETERS:    NONE
                        01B4    710 ;
                        01B4    711 ; IMPLICIT OUTPUTS:
                        01B4    712 ;
                        01B4    713 ;       THE DS$GL_TIMQFL QUEUE IS EMPTY
                        01B4    714 ;
                        01B4    715 ; SIDE EFFECTS:         NONE
                        01B4    716 ;
                        01B4    717 ;--
                        01B4    718
                        01B4    719 DSR$SETIMR_INI::
     50   0000'CF   DE  01B4    720          MOVAL   W^DS$GL_TIMQFL,R0       ; GET QUEUE ADDRESS
          60   50   D0  01B9    721          MOVL    R0,(R0)                 ; Flink to self
       04 A0   50   D0  01BC    722          MOVL    R0,4(R0)                ; Blink to self
50   00000000'EF   7D  01C0    723          MOVQ    EXE$GQ_SYSTIME,R0       ; Expected completion time
55   00000024'EF   DE  01C7    724          MOVAL   ONESECTIM,R5            ; Point to one/sec timer
               01   10  01CE    725          BSBB    EXE$INSTIMQ             ; Insert in timer queue
                    05  01D0    726          RSB                             ; RETURN
```

```
                        01D1    728 ;++
                        01D1    729 ; FUNCTIONAL DESCRIPTION:
                        01D1    730 ;
                        01D1    731 ;       This routine is used to place entries in the time queue
                        01D1    732 ;       The queue is ordered by completion time. the soonest request
                        01D1    733 ;       being the first in the queue.
                        01D1    734 ;
                        01D1    735 ; CALLING SEQUCENE:
                        01D1    736 ;
                        01D1    737 ;       BSBW    EXE$INSTIMQ
                        01D1    738 ;
                        01D1    739 ;
                        01D1    740 ; INPUT PARAMETERS:
                        01D1    741 ;
                        01D1    742 ;       R0/R1           64 bit completion time
                        01D1    743 ;       R5              Address of TQE
                        01D1    744 ;
                        01D1    745 ;
                        01D1    746 ; IMPLICIT INPUTS:      NONE
                        01D1    747 ;
                        01D1    748 ; OUTPUT PARAMETERS:    NONE
                        01D1    749 ;
                        01D1    750 ; IMPLICIT OUTPUTS:     NONE
                        01D1    751 ;
                        01D1    752 ;--
                        01D1    753
                        01D1    754 EXE$INSTIMQ::
                        01D1    755
      18 A5    50   7D  01D1    756             MOVQ    R0,TQE$Q_TIME(R5)       ; Insert completion time in TQE
                        01D5    757             DSBINT  #IPL$_TIMER             ; Disable to timer level
54    00000000'EF      DE  01DB    758             MOVAL   DS$GL_TIMQFL,R4         ; Point to queue head
         53    54      D0  01E2    759             MOVL    R4,R3                   ; Copy start of queue
      53   04 A3      D0  01E5    760 50$:        MOVL    TQE$L_TQBL(R3),R3       ; Link to next entry
         54    53      D1  01E9    761             CMPL    R3,R4                   ; End of queue?
               0E      13  01EC    762             BEQL    60$                     ; Yes insert it here
      1C A3    51      D1  01EE    763             CMPL    R1,TQE$Q_TIME+4(R3)     ; Compare High order parts of time
               F1      1F  01F2    764             BLSSU   50$                     ; If LSSU new entry more imminent
               06      1A  01F4    765             BGTRU   60$                     ; If GTRU new entry less imminent
      18 A3    50      D1  01F6    766             CMPL    R0,TQE$Q_TIME(R3)       ; Compare low order part of time
               E9      1F  01FA    767             BLSSU   50$                     ; If LSSU new entry more imminent
         63    65      0E  01FC    768 60$:        INSQUE  (R5), (R3)              ; Insert new entry                      [18]
                           01FF    769             ENBINT
                     05    0202    770             RSB
```

```
                        0203    772          .SBTTL  EXE$Setimr -    SET TIMER SYSTEM SERVICE.
                        0203    773 ;++
                        0203    774 ; FUNCTIONAL DESCRIPTION:
                        0203    775 ;
                        0203    776 ;       THE SET TIMER SYSTEM SERVICE ALLOWS A PROCESS TO CAUSE THE SETTING OF
                        0203    777 ;       AN EVENT FLAG AND THE ISSUANCE OF AN AST AT SOME FUTURE TIME.  THE
                        0203    778 ;       TIME FOR THE EVENT CAN BE SPECIFIED AS AN OFFSET FROM THE CURRENT
                        0203    779 ;       TIME, THAT IS, DELTA TIME.
                        0203    780 ;
                        0203    781 ; CALLING SEQUENCE:
                        0203    782 ;
                        0203    783 ;       STARLET PROCEDURE CALL.
                        0203    784 ;
                        0203    785 ; INPUT PARAMETERS:
                        0203    786 ;
                        0203    787 ;       EFN     = EVENT FLAG NUMBER OF THE EVENT FLAG TO SET WHEN THE
                        0203    788 ;                 TIME INTERVAL EXPIRES.  IF NOT SPECIFIED, IT DEFAULTS
                        0203    789 ;                 TO 0.
                        0203    790 ;       DAYTIM  = ADDRESS OF THE QUADWORD EXPIRATION TIME.  IF THE TIME
                        0203    791 ;                 VALUE IS NEGATIVE, IT INDICATES AN OFFSET (DELTA TIME)
                        0203    792 ;                 FROM THE CURRENT TIME.
                        0203    793 ;       ASTADR  = ADDRESS OF THE ENTRY MASK OF AN AST ROUTINE TO BE
                        0203    794 ;                 CALLED WHEN THE TIME INTERVAL EXPIRES.  IF 0, IT
                        0203    795 ;                 INDICATES NO AST IS TO BE QUEUED.
                        0203    796 ;       REQIDT  = COPIED TO ASTPRM ON AST DELIVERY.
                        0203    797 ;
                        0203    798 ; IMPLICIT INPUTS:      NONE
                        0203    799 ;
                        0203    800 ; OUTPUT PARAMETERS:    NONE
                        0203    801 ;
                        0203    802 ; IMPLICIT OUTPUTS:     NONE
                        0203    803 ;
                        0203    804 ; COMPLETION CODES:     NONE
                        0203    805 ;
                        0203    806 ;       SS$_EXQUOTA     = QUOTA FOR TIMER REQUESTS EXCEEDED.
                        0203    807 ;       SS$_NOTIMP      = FUNCTION NOT IMPLEMENTED IN STAND-ALONE.
                        0203    808 ;       SS$_NORMAL      = NORMAL RETURN.
                        0203    809 ;
                        0203    810 ; SIDE EFFECTS:         NONE
                        0203    811 ;--
```

```
                        003C  0203   813 .ENTRY  EXE$SETIMR,^M<R2,R3,R4,R5>
                              0205   814
                              0205   815          CMK     SETIMR                          ; Execute code in KERNEL mode
                        04    0214   816          RET                                     ; All done, return
                              0215   817
                              0215   818 ;+
                              0215   819 ; KERNEL mode routine to SETIMR
                              0215   820 ;-
                              0215   821 CMK$SETIMR::
                 50    1C     D0     0215   822          MOVL    S^#SS$_EXQUOTA,R0       ; PRESUME QUOTA EXCEEDED
                     FDE5'    30     0218   823          BSBW    W^KB_CHECK
        00000000'EF    11     E0     021B   824          BBS     #DS$V_TIMRON, DS$GL_FLAGS -
                       64            0222   825                  ,50$                    ; Can't SETIMR if TIMER owned
                              0223   826
                              C223   827          $CLREF_G  (AP)                          ; CLEAR THE EVENT FLAG.
                    5A 50     E9     022A   828          BLBC    R0,50$                  ; Exit if CLREF failed
                              022D   829
        00000000'EF    40 8F   8A     022D   830          BICB2   #DS$M_ABRTFLG, -        ; CLEAR ABORTWAIT FLAG.
                              0235   831                  DS$GL_FLAGS
                              0235   832
     50    006600B8 8F     D0     0235   833          MOVL    #DS$_.PL2HI, R0         ; ANTICIPATE IPL TOO HIGH
                       51     DC     023C   834          MOVPSL  R1                      ; GET PROCESSOR STATUS.
                 05    10     ED     023E   835          CMPZV   #PSL$V_IPL,#PSL$S_IPL,- ; CHECK IPL.
                 02    51            0241   836                  R1, #2
                       42     18     0243   837          BGEQ    50$                     ; If GEQ, IPL is too high
     50    006600B0 8F     D0     0245   838          MOVL    #DS$_NOTIMP,R0          ; NOT IMPLEMENTED FOR ABSOLUTE TIME
      51    08 BC    7D     024C   839          MOVQ    @SETIMR$_DAYTIM(AP),R1  ; TEST SIGN BIT OF QUADWORD
                       35     14     0250   840          BGTR    50$                     ; Branch to exit if invalid delta time
                              0252   841 ;
                              0252   842 ; ALLOCATE SPACE AND BUILD A TIMER QUEUE ENTRY FOR THIS TIME REQUEST
                              0252   843 ;
                     FDAB'    30     0252   844          BSBW    EXE$ALLOCTQE            ; RETURNS R2=ADDR OF BLOCK IF SUCCEEDS
                  2F 50       E9     0255   845          BLBC    R0,50$                  ; Exit if no space for TQE
                55    52      D0     0258   846          MOVL    R2,R5                   ; Copy address of TQE
                   0B A5      94     025B   847          CLRB    TQE$B_RQTYPE(R5)        ; CLEAR AST CONTROL BLOCK FLAGS
 0C A5    00000000'8F        D0     025E   848          MOVL    #IOC$K_DIAGPID,TQE$L_PID(R5) ; SET PID
    10 A5    0C AC     7D     0266   849          MOVQ    SETIMR$_ASTADR(AP), -
                              026B   850                  TQE$L_AST(R5)           ; COPY AST ADDRESS AND PARAMETER
    29 A5    04 AC     90     026B   851          MOVB    SETIMR$_EFN(AP), -      ; COPY EVENT FLAG NUMBER TO TQE
                              0270   852                  TQE$B_EFN(R5)
       7E    08 BC     7D     0270   853          MOVQ    @SETIMR$_DAYTIM(AP),-(SP) ; GET DURATION TIME
  50    00000000'EF     7D     0274   854          MOVQ    EXE$GQ_SYSTIME,R0       ; GET CURRENT TIME
          50    8E      C2     0278   855          SUBL2   (SP)+,R0                ; Current -(-DELTA)=Current+DELTA
          51    8E      D9     027E   856          SBWC    (SP)+,R1                ; Same with high order
             FF4D       30     0281   857          BSBW    EXE$INSTIMQ             ; Insert it in the timer queue
          50    01      D0     0284   858          MOVL    S^#SS$_NORMAL,R0        ; RETURN SUCCESS
                              0287   859                                          ; FALL IN TO EXIT
                        02    0287   860 50$:     REI                             ; Return to calling mode
```

```
                     0288   862          .SBTTL  DSI$TimSrv -    INTERVAL TIMER INTERRUPT SERVICE ROUTINE.
                     0288   863 ;++
                     0288   864 ; FUNCTIONAL DESCRIPTION:
                     0288   865 ;
                     0288   866 ;       INTERVAL TIMER INTERRUPT SERVICE ROUTINE
                     0288   867 ;
                     0288   868 ; CALLING SEQUENCE:
                     0288   869 ;
                     0288   870 ;       INTERRUPT
                     0288   871 ;
                     0288   872 ; IMPLICIT INPUTS:
                     0288   873 ;
                     0288   874 ;       NONE
                     0288   875 ;
                     0288   876 ; IMPLICIT OUTPUTS:
                     0288   877 ;
                     0288   878 ;       NONE
                     0288   879 ;
                     0288   880 ; SIDE EFFECTS:
                     0288   881 ;
                     0288   882 ;       NONE
                     0288   883 ;--
```

```
                                    0288    885          .ALIGN  LONG
                                    0288    886 DSI$TIMSRV::
                              50    DD  0288    887          PUSHL   R0                          ; Save a register
                                    028A    888
    00000000'EF    000186A0 8F    C0  028A    889          ADDL2   #TICK_NS, EXE$GQ_SYSTIME ; INCREMENT SYSTIME
          00000004'EF    00    D8  0295    890          ADWC    #0, EXE$GQ_SYSTIME+4    ; AND HIGH HALF
                                    029C    891
            09 00000000'EF    E8  029C    892          Blbs    L^DS$GB_Stopping_The_Timer, -   ; Don't start the interval      [22]
                                    02A3    893                    3$                       ; . . . timer up again if we are in the [25]
                                    02A3    894                                             ; . . . the process of stopping it - in [22]
                                    02A3    895                                             ; . . . the DSX$SETVEC routine, because of
                                    02A3    896                                             ; . . . the asynchronous nature of the  [22]
                                    02A3    897                                             ; . . . VAX 11/730 console instructions [22]
                                    02A3    898
          18    800000C1 8F    DA  C2A3    899          MTPR    #ICCS$M_RESET,#PR$_ICCS ; Resetting the timer
                              07    11  02AA    900          BRB     5$                          ; Normal flow                   [25]
                                    02AC    901 ;+
                                    02AC    902 ;       CHECK TIMER QUEUE FOR DUE REQUESTS
                                    02AC    903 ;-
                                    02AC    904
          18    80000080 8F    DA  02AC    905 3$:      MTPR    #ICCS$M_ERR+ICCS$M_INT, -
                                    02B3    906                    #PR$_ICCS                ; Clear the interrupt to avoid race     [25]
                                    02B3    907                                             ; with DSX$SETVEC                       [25]
          50    00000000'EF    D0  02B3    908 5$:      MOVL    DS$GL_TIMQFL,R0             ; Get address of timer queue entry      [22]
          00000000'8F    50    D1  02BA    909          CMPL    R0,#DS$GL_TIMQFL            ; Anything in queue?
                              21    13  02C1    910          BEQL    20$                         ; Branch if not
    00000004'EF    1C A0    D1  02C3    911          CMPL    TQE$Q_TIME+4(R0),EXE$GQ_SYSTIME+4 ; Compare high order
                              0C    1F  02CB    912          BLSSU   10$                         ; If LSSU entry due
                              15    1A  02CD    913          BGTRU   20$                         ; If GTRU entry not due
    00000000'EF    18 A0    D1  02CF    914          CMPL    TQE$Q_TIME(R0),EXE$GQ_SYSTIME ; Compare low order
                              0B    1A  02D7    915          BGTRU   20$                         ; IF GTRU entry not due
                                    02D9    916
                                    02D9    917 10$:     SOFTINT #IPL$_TIMER                 ; Entry is due dispatch on timer level
    00 00000000'EF    07    E2  02DC    918          BBSS    #IPL$_TIMER,DS$GA_SOFTINT,20$; Set level 7 interrupt expected
                                    02E4    919
                              50 8ED0   02E4    920 20$:     POPL    R0                          ; Restore R0
                                 02   02E7    921          REI
```

```
                     02E8    923          .SBTTL  DSI$TIMER -     Level 7 timer interrupt service
                     02E8    924 ;++
                     02E8    925 ; FUNCTIONAL DESCRIPTION:
                     02E8    926 ;
                     02E8    927 ;       This interrupt service routine is called from a software level 7
                     02E8    928 ;       timer interrupt.  It is used to process timer queue entries that
                     02E8    929 ;       are due and to check for hung qio devices
                     02E8    930 ;
                     02E8    931 ; CALLING SEQUENCE:
                     02E8    932 ;
                     02E8    933 ;       Interrupt service entry
                     02E8    934 ;
                     02E8    935 ;--
```

```
                                02E8   937          .ALIGN  LONG
                                02E8   938 DSI$TIMER::
                          3F    BB  02E8   939          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save a few
                                02EA   940
                                02EA   941 ;+
                                02EA   942 ;       CHECK TIMER QUEUE FOR DUE REQUESTS
                                02EA   943 ;-
                                02EA   944
                                02EA   945 10$:    SETIPL  #IPL$_HWCLK             ; Prevent interruptions by clock
     55   00000000'EF          D0  02ED   946          MOVL    DS$GL_TIMQFL,R5         ; Get address of timer queue entry
     00000000'8F    55        D1  02F4   947          CMPL    R5,#DS$GL_TIMQFL        ; Anything in queue?
                    03        12  02FB   948          BNEQ    15$                     ; Branch if something in queue
                  007A        31  02FD   949          BRW     40$                     ; Exit if empty
                              0300   950 15$:
 00000004'EF    1C A5          D1  C300   951          CMPL    TQE$Q_TIME+4(R5),EXE$GQ_SYSTIME+4 ; Compare high order
                    0C        1F  0308   952          BLSSU   20$                     ; If LSSU entry due
                    6E        1A  030A   953          BGTRU   40$                     ; If GTRU entry not due
 00000000'EF    18 A5          D1  030C   954          CMPL    TQE$Q_TIME(R5),EXE$GQ_SYSTIME ; Compare low order
                    64        1A  0314   955          BGTRU   40$                     ; IF GTRU entry not due
                              0316   956 ;+
                              0316   957 ; Remove entry from queue
                              0316   958 ;-
                              0316   959
                50    65      0F  0316   960 20$:    REMQUE  (R5), R0                ; Remove queue entry                    [18]
                              0319   961          SETIPL  #IPL$_TIMER             ; Back to our level
 50   0B A5    02    00      EF  031C   962          EXTZV   #0,#2,TQE$B_RQTYPE(R5),R0; Get request type
                              0322   963          CASE    R0,<25$,35$>            ; Dispatch on type
                              032A   964          ERRSUP_S
                    AD    11  0338   965          BRB     10$                     ; Check again
```

```
                                033D    967 ;+
                                033D    968 ; Process timer entry that queues an AST or sets an event flag
                                033D    969 ;-
                                033D    970
          7E    29 A5    9A   033D    971 25$:    MOVZBL  TQE$B_EFN(R5),-(SP)     ; SPECIFY EVENT FLAG NUMBER
00000000'9F       01    FB   0341    972          CALLS   #1,@#SYS$SETEF          ; AND GO SET IT
                10 A5    D5   0348    973          TSTL    TQE$L_AST(R5)           ; AST ADDRESS SPECIFIED ?
                   07    13   034B    974          BEQL    30$                     ; IF EQL, NO AST SPECIFIED
                                034D    975
                   52    D4   034D    976          CLRL    R2                      ; PRIORITY INCREMENT CLASS
                 FCAE'   30   034F    977          BSBW    SCH$QAST                ; GO SCHEDULE AN AST FOR THIS TIMER
                   96    11   0352    978          BRB     10$                     ; GO BACK FOR NEXT ENTRY
                                0354    979
              50    55    D0   0354    980 30$:    MOVL    R5,R0                   ; POINT TO CURRENT ENTRY
                 FCA6'   30   0357    981          BSBW    EXE$DEANONPAGED         ;  AND RETURN IT TO FREE MEMORY POOL
                   8E    11   035A    982          BRB     10$                     ; BACK TO LOOK AT NEXT ENTRY
                                035C    983
                                035C    984 ;+
                                035C    985 ; call timer driven system routine
                                035C    986 ;-
                                035C    987
          53    10 A5    7D   035C    988 35$:    MOVQ    TQE$L_FR3(R5),R3        ; Restore subroutine context
                0C B5    16   0360    989          JSB     @TQE$L_FPC(R5)          ; Call routine
       0F 0B A5    02    E1   0363    990          BBC     #TQE$V_REPEAT,TQE$B_RQTYPE(R5),38$; if not repeatable
          50    20 A5    7D   0368    991          MOVQ    TQE$Q_DELTA(R5),R0      ; Get delta time
          50    18 A5    C0   036C    992          ADDL    TQE$Q_TIME(R5),R0       ; Add low order parts
          51    1C A5    D8   0370    993          ADWC    TQE$Q_TIME+4(R5),R1     ; Add high order parts w/ carry
                 FE5A    30   0374    994          BSBW    EXE$INSTIMQ             ; re-insert in the timer queue
                                0377    995 38$:
                 FF70    31   0377    996          BRW     10$                     ; Look in queue again
                                037A    997
                                037A    998 40$:
                                037A    999          SETIPL  #IPL$_TIMER             ; Back to our level
                                037D   1000 DSI$TIMER_X:
          3F    BA              037D   1001          POPR    #^M<R0,R1,R2,R3,R4,R5>
                02              037F   1002          REI                             ; Exit from timer service
```

```
                                0380  1004 ;+
                                0380  1005 ; This routine is called each second to update the time
                                0380  1006 ; and check for timed out UCB's
                                0380  1007 ;-
                                0380  1008
                                0380  1009 ONE_SECOND:
     00000000'EF    D6          0380  1010          INCL    EXE$GL_ABSTIM           ; Update time
                                0386  1011
         7E    55  7D          0386  1012          MOVQ    R5,-(SP)                ; Save registers
56   00000000'EF    DE          0389  1013          MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK, R6 ; Locate first DDB              [17]
         56    66  D0          0390  1014 10$:     MOVL    DDB$L_LINK(R6),R6       ; LOCATE NEXT DDB
               3F  13          0393  1015          BEQL    60$                     ; IF EQL, NO MORE DDB'S
                                0395  1016                                          ;
          55   D8 A6  DE       0395  1017 20$:     MOVAL   DDB$L_UCB-UCB$L_LINK(R6),R5     ; LOCATE FIRST UCB
                                0399  1018                                          ;
          55   2C A5  D0       0399  1019 30$:     MOVL    UCB$L_LINK(R5),R5       ; LOCATE NEXT UCB
               F1  13          039D  1020          BEQL    10$                     ; IF EQL, NO MORE UCB'S THIS DDB
                                039F  1021 ;
                                039F  1022 ; AT THIS POINT A UCB HAS BEEN LOCATED
                                039F  1023 ;
                                039F  1024
                                039F  1025          SETIPL  #IPL$_POWER                     ; Raise to Powerfail level
      28 58 A5    00  E1        03A2  1026          BBC     #UCB$V_TIM,UCB$W_STS(R5),50$    ; IF NOT WAITING, GET NEXT UCB
00000000'EF    5C A5  D1        03A7  1027          CMPL    UCB$L_DUETIM(R5),EXE$GL_ABSTIM  ; CHECK DUE TIME FOR TIMED OUT
               1E  14           03AF  1028          BGTR    50$                     ; IF GTR, LET IT WAIT LONGER
                                03B1  1029
          58 A5  03  AA         03B1  1030          BICW    #UCB$M_INT!UCB$M_TIM, -
                                03B5  1031                  UCB$W_STS(R5)           ; Clear interrupt and timer
58 A5   0040  8F  A8            03B5  1032          BISW    #UCB$M_TIMOUT,UCB$W_STS(R5)     ; SET TIMED OUT STATUS BIT
                                03BB  1033          SETIPL  UCB$B_DIPL(R5)          ; Set IPL to device level
       53   10 A5  7D           03BF  1034          MOVQ    UCB$L_FR3(R5),R3        ; RETREIVE FORK CONTEXT
       52   0C A5  D0           03C3  1035          MOVL    UCB$L_FPC(R5),R2        ; Get sved pc
          7E   72  32           03C7  1036          CVTWL   -(R2),-(SP)             ; Get offset to exception routine
          52   8E  C0           03CA  1037          ADDL    (SP)+,R2                ; Address of sxception routine
               62  16           03CD  1038          JSB     (R2)                    ; RE-ACTIVATE I/O REQUEST
                                03CF  1039 50$:     SETIPL  #IPL$_TIMER             ; Restore to timer IPL
               C5  11           03D2  1040          BRB     30$                     ; GO LOCATE NEXT UCB
                                03D4  1041
          55   8E  7D           03D4  1042 60$:     MOVQ    (SP)+,R5                ; Restore
                    05          03D7  1043          RSB                             ; Return from whence we came
```

```
                     1045          .SBTTL  CLK$RE_INIT -   SYSTEM TIMER INITIALIZATION SUBROUTINE
                     1046 ;++
                     1047 ; FUNCTIONAL DESCRIPTION:
                     1048 ;
                     1049 ;       INITIALIZES THE INTERVAL TIMER FOR 10 MILLISECOND TICKS
                     1050 ;       AND SYNCHRONIZES THE SYSTEM TIME TO THE TIME OF YEAR CLOCK
                     1051 ;
                     1052 ; CALLING SEQUENCE:
                     1053 ;
                     1054 ;       BSBW    CLK$RE_INIT
                     1055 ;
                     1056 ; IMPLICIT INPUTS:
                     1057 ;
                     1058 ;       NONE
                     1059 ;
                     1060 ; IMPLICIT OUTPUTS:
                     1061 ;
                     1062 ;       NONE
                     1063 ;
                     1064 ; COMPLETION CODES:
                     1065 ;
                     1066 ;       NONE
                     1067 ;
                     1068 ; SIDE EFFECTS:
                     1069 ;
                     1070 ;       INTERVAL TIMER IS LEFT RUNNING (INTERRUPT ENABLED)
                     1071 ;       NEXT INTERVAL REGISTER IS SET TO 10 MILLISECONDS
                     1072 ;
                     1073 ;       IF PR$_TODR VALID,
                     1074 ;           EXE$GQ_SYSTIME      =  CURRENT DATE/TIME (64-BIT FORMAT)
                     1075 ;--
```

```
                                        03D8  1077 CLK$RE_INIT::
                                        03D8  1078
                                        03D8  1079 ;+
                                        03D8  1080 ; SETUP AND START INTERVAL TIMER
                                        03D8  1081 ;-
                                        03D8  1082
          18   80000080 8F    DA       03D8  1083          MTPR    #ICCS$M_ERR + -          ; CLEAR ERROR,
                                        03DF  1084                   ICCS$M_INT, -          ; CLEAR DONE,
                                        03DF  1085                         #PR$_ICCS        ; AND STOP INTERVAL TIMER
          19   FFFFD8F0 8F    DA       03DF  1086          MTPR    #-TICK_US, #PR$_NICR     ; SET NEXT INTERVAL.
          18   00000051 8F    DA       03E6  1087          MTPR    #ICCS$M_IE + -           ; ENABLE INTERRUPT,
                                        03ED  1088                   ICCS$M_XFER + -        ; LOAD,
                                        03ED  1089                   ICCS$M_RUN, -          ; AND RESTART
                                        03ED  1090                         #PR$_ICCS        ; THE INTERVAL CLOCK.
                                        03ED  1091
                                        03ED  1092 ;+
                                        03ED  1093 ; SET SYSTIME FROM TOD IF TOD IS VALID
                                        03ED  1094 ;-
                                        03ED  1095
               00000000'EF   16         03ED  1096          JSB     ONLY_CLOCK_INIT          ; GET TIME OF YEAR IN R0 AND DO          [30]
                                        03F3  1097                                           ; CPU-SPECIFIC STUFF TO TODR             [30]
          50   10000000 8F   C2         03F3  1098          SUBL2   #TODR_BIAS, R0           ; REMOVE VALIDATION BIAS
                        29   1F         03FA  1099          BLSSU   CLK$RE_SYNC_X            ; IF LSSU, TODR WAS N.G.
                                        03FC  1100
               50   01   CA             03FC  1101          BICL    #1,R0                    ; CLEAR LOW BIT
          50   50   FF 8F   9C         03FF  1102          ROTL    #-1,R0,R0                ; DIVIDE BY 2 UNSIGNED
    50    00   50   00030D40 8F   7A    0404  1103          EMUL    #2*TICK_NS, R0, -        ; CONVERT TIME OF YEAR TO
                                        040D  1104                    #0, R0                ;   SYSTIME FORMAT (10MS TO 100NS)
               50   0008'CF   C0         040D  1105          ADDL2   W^DS$GQ_CURYEAR, R0      ; ADJUST TO CURRENT TIME/DATE
               51   000C'CF   D8         0412  1106          ADWC    W^DS$GQ_CURYEAR+4, R1    ;   (ALL 64 BITS)
          00000000'EF   50   7D         0417  1107          MOVQ    R0, EXE$GQ_SYSTIME       ; Set system date/time
          0000003C'EF   50   7D         041E  1108          MOVQ    R0, ONESECTIM+TQE$Q_TIME ; Reset 1/second timer expiration time
                                        0425  1109
                                        0425  1110 CLK$RE_SYNC_X:
                              05        0425  1111          RSB
                                        0426  1112
                                        0426  1113
                                        0426  1114          .END
```

```
$$ARGS                          = 00000004     D
$$T1                            = 00000000     D
$ER                             = 00000002     D
$MODULE                           00000000 R   D  03
BIT...                          = 0000001A     D
CANTIM$_ACMODE                  = 00000008     D
CANTIM$_NARGS                   = 00000002     D
CANTIM$_REQIDT                  = 00000004     D
CLK$RE_INIT                       000003D8 RG  D  04
CLK$RE_SYNC_X                     00000425 R   D  04
CLOCK_IPL                       = 00000018     D
CMK$CANTIM                        0000017C RG  D  04
CMK$HIBER                         00000151 RG  D  04
CMK$HIBER_X                       00000166 R   D  04
CMK$SETIMR                        00000215 RG  D  04
CMK$                            = 00000004     D
CMK$_ASTEXIT                    = 00000000     D
CMK$_CANTIM                     = 0000000B     D
CMK$_HIBER                      = 00000007     D
CMK$_INITSCB                    = 00000008     D
CMK$_LAST                       = 0000000E     D
CMK$_MMENABLE                   = 00000003     D
CMK$_RCONSOLE                   = 00000001     D
CMK$_REFRESH                    = 00000005     D
CMK$_SCHDWK                     = 00000006     D
CMK$_SETIMR                     = 0000000C     D
CMK$_SETIPL                     = 00000009     D
CMK$_SETPRT                     = 0000000D     D
CMK$_SGIPR                      = 0000000A     D
CMK$_TCONSOLE                   = 00000002     D
DDB$B_ACPCLASS                    00000013     D
DDB$B_TYPE                        0000000A     D
DDB$C_LENGTH                      00000034     D
DDB$K_LENGTH                      00000034     D
DDB$L_ACPD                        00000010     D
DDB$L_DDT                         0000000C     D
DDB$L_LINK                        00000000     D
DDB$L_UCB                         00000004     D
DDB$T_DRVNAME                     00000024     D
DDB$T_NAME                        00000014     D
DDB$W_SIZE                        00000008     D
DS$CANWAIT                        ********   X    02
DS$GA_SOFTINT                     ********   X    04
DS$GB_STOPPING_THE_TIMER          ********   X    04
DS$GK_USOVR                       ********   X    04
DS$GL_FLAGS                       ********   X    04
DS$GL_TIMQFL                      00000000 R   D  02
DS$GQ_CURYEAR                     00000008 RG  D  02
DS$GT_NEWYEAR                     00000010 RG  D  02
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_NYSIZ                      = 00000014   G D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DS$WAITUS                         ********   X    04
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
```

```
DS$_CHMK                = 006600E0   D
DS$_DEVNAME             = 00660108   D
DS$_ERROR               = 00660002   D
DS$_FHWE                = 00660068   D
DS$_FRAGBUF             = 00660080   D
DS$_ICBUSY              = 006600C8   D
DS$_ICERR               = 006600C0   D
DS$_IHWE                = 00660060   D
DS$_ILLCHAR             = 00660018   D
DS$_ILLPAGCNT           = 00660078   D
DS$_ILLUNIT             = 00660100   D
DS$_INSFMEM             = 00660050   D
DS$_IPL2HI              = 006600B8   D
DS$_IVADDR              = 00660040   D
DS$_IVVECT              = 00660038   D
DS$_KRNLSTK             = 00660090   D
DS$_LOGIC               = 00660070   D
DS$_MCHK                = 00660088   D
DS$_MMOFF               = 00660058   D
DS$_NEEDUNIT            = 006600F8   D
DS$_NODE                = 00660128   D
DS$_NOPCS               = 00660110   D
DS$_NORMAL              = 00660001   D
DS$_NOSUPPORT           = 006600B1   D
DS$_NOTDON              = 00660030   D
DS$_NOTIMP              = 006600B0   D
DS$_NULLSTR             = 00660010   D
DS$_OVERFLOW            = 00660008   D
DS$_POWER               = 00660098   D
DS$_PROGERR             = 00660020   D
DS$_SEVERE              = 00660004   D
DS$_TRANSL              = 006600A0   D
DS$_TRUNCATE            = 00660028   D
DS$_UNEXPINT            = 006600D8   D
DS$_VASFULL             = 00660048   D
DS$_WARNING             = 00660000   D
DSA$AL_APTMAIL            0000FE00   D
DSA$AT_APTTXT             0000FA00   D
DSA$GL_APTCOM             0000FE04   D
DSA$GL_DEVLEN             0000FE58   D
DSA$GL_ERRNO              0000FE44   D
DSA$GL_EVENT              0000FE48   D
DSA$GL_FLAGS              0000FE00   D
DSA$GL_MSGTYP             0000FE40   D
DSA$GL_PASSES             0000FE08   D
DSA$GL_PASSNO             0000FE54   D
DSA$GL_SECTNO             0000FE10   D
DSA$GL_SID                0000FE14   D
DSA$GL_SUBTNO             0000FE4C   D
DSA$GL_TESTNO             0000FE50   D
DSA$GL_UNITS              0000FE0C   D
DSA$GQ_MSGPTR             0000FE68   D
DSA$GT_DEVNAM             0000FE5C   D
DSI$TIMER               = 000002E8   RG D   04
DSI$TIMER_X             = 0000037D   R  D   04
DSI$TIMSRV              = 00000288   RG D   04
DSR$SETIMR_INI            000001B4   RG D   04
DSX$CANWAIT               00000000   RG D   04
DSX$WAITMS                00000027   RG D   04
DSX$WAITMS_X              000000AF   R  D   04
DSX$WAITUS                000000B0   RG D   04
DSX$WAITUS_USOVR          000000CB   RG D   04
DSX$WAITUS_X              00000130   R  D   04
DS_ERRSUP                 ********     X   04
DYN$C_TQE               = 0000000F     D
EXE$ALLOCTQE              ********     X   04
EXE$CANTIM                00000167   RG D   04
EXE$DEANONPAGED           ********     X   04
EXE$GL_ABSTIM             ********     X   04
EXE$GQ_SYSTIME            ********     X   04
EXE$HIBER                 00000131   RG D   04
EXE$HIBER_X               00000150   R  D   04
EXE$INSTIMQ               000001D1   RG D   04
EXE$SETIMR                00000203   RG D   04
EXE$WAKE                  0000001B   RG D   04
GETTIM$_NARGS           = 00000001     D
GETTIM$_TIMADR          = 00000004     D
ICCS$M_ERR              = 80000000     D
ICCS$M_IE               = 00000040     D
ICCS$M_INT              = 00000080     D
ICCS$M_RESET            = 800000C1     D
ICCS$M_RUN              = 00000001     D
ICCS$M_XFER             = 00000010     D
IOC$GL_DEVLIST            ********     X   04
IOC$K_DIAGPID             ********     X   04
IPL$_HWCLK              = 00000018     D
IPL$_POWER              = 0000001F     D
IPL$_TIMER              = 00000007     D
KB_CHECK                  ********     X   04
MSTIME                    00000060   R  D   02
ONESECTIM                 00000024   R  D   02
ONE_SECOND                00000380   R  D   04
ONLY_CLOCK_INIT           ********     X   04
PR$_ICCS                = 00000018     D
PR$_ICR                 = 0000001A     D
PR$_IPL                 = 00000012     D
PR$_NICR                = 00000019     D
PR$_SIRR                = 00000014     D
PSL$S_IPL               = 00000005     D
PSL$V_IPL               = 00000010     D
PSL$V_IS                = 0000001A     D
SCB$L_ACCESS              00000020     D
SCB$L_ARITH               00000034     D
SCB$L_BREAK               0000002C     D
SCB$L_CHME                00000044     D
SCB$L_CHMK                00000040     D
SCB$L_CHMS                00000048     D
SCB$L_CHMU                0000004C     D
SCB$L_COMPAT              00000030     D
SCB$L_KNLSTK              00000008     D
SCB$L_MACHCHK             00000004     D
SCB$L_OPCCUS              00000014     D
SCB$L_OPCDEC              00000010     D
SCB$L_POWER               0000000C     D
```

```
SCB$L_RADRMOD                 0000001C    D
SCB$L_ROPRAND                 00000018    D
SCB$L_RXDB                    000000F8    D
SCB$L_SFTLVL1                 00000084    D
SCB$L_SFTLVL10                000000A8    D
SCB$L_SFTLVL11                000000AC    D
SCB$L_SFTLVL12                000000B0    D
SCB$L_SFTLVL13                000000B4    D
SCB$L_SFTLVL14                000000B8    D
SCB$L_SFTLVL15                000000BC    D
SCB$L_SFTLVL2                 00000088    D
SCB$L_SFTLVL3                 0000008C    D
SCB$L_SFTLVL4                 00000090    D
SCB$L_SFTLVL5                 00000094    D
SCB$L_SFTLVL6                 00000098    D
SCB$L_SFTLVL7                 0000009C    D
SCB$L_SFTLVL8                 000000A0    D
SCB$L_SFTLVL9                 000000A4    D
SCB$L_TBIT                    00000028    D
SCB$L_TIMER                   000000C0    D
SCB$L_TRANSL                  00000024    D
SCB$L_TXDB                    000000FC    D
SCB$L_ZERO                    00000000    D
SCH$QAST                      ********  X   04
SEC_TICK                =     00000064  G D
SETIMR$_ASTADR          =     0000000C    D
SETIMR$_DAYTIM          =     00000008    D
SETIMR$_EFN             =     00000004    D
SETIMR$_NARGS           =     00000004    D
SETIMR$_REQIDT          =     00000010    D
SIZ...                  =     00000001    D
SS$_EXQUOTA             =     0000001C    D
SS$_NORMAL              =     00000001    D
SYS$CANTIM                    ********  GX  04
SYS$CLREF                     ********  GX  04
SYS$GETTIM                    ********  GX  04
SYS$HIBER                     ********  GX  04
SYS$SETEF                     ********   X  04
SYS$SETIMR                    ********  GX  04
SYS$WAKE                      ********   X  04
TICK_NS                 =     000186A0    D
TICK_US                 =     00002710    D
TODR_BIAS               =     10000000    D
TQE$B_EFN               =     00000029    D
TQE$B_RQTYPE            =     0000000B    D
TQE$C_SSREPT            =     00000005    D
TQE$L_AST               =     00000010    D
TQE$L_ASTPRM            =     00000014    D
TQE$L_FPC               =     0000000C    D
TQE$L_FR3               =     00000010    D
TQE$L_PID               =     0000000C    D
TQE$L_TQBL              =     00000004    D
TQE$L_TQFL              =     00000000    D
TQE$Q_DELTA             =     00000020    D
TQE$Q_TIME              =     00000018    D
TQE$V_REPEAT            =     00000002    D
UCB$B_AMOD                    00000053    D
UCB$B_CEX                     00000077    D
UCB$B_CMI                     0000004A    D
UCB$B_CM2                     0000004B    D
UCB$B_DEVCLASS                00000038    D
UCB$B_DEVTYPE                 00000039    D
UCB$B_DIPL                    00000052    D
UCB$B_DX_SCTCNT               000000A6    D
UCB$B_ERTCNT                  00000070    D
UCB$B_ERTMAX                  00000071    D
UCB$B_FEX                     00000076    D
UCB$B_FIPL                    0000000B    D
UCB$B_LOCSRV                  0000003C    D
UCB$B_OFFNDX                  00000094    D
UCB$B_OFFRTC                  00000095    D
UCB$B_REMSRV                  0000003D    D
UCB$B_SECTORS                 0000003C    D
UCB$B_SLAVE                   00000074    D
UCB$B_SPR                     00000075    D
UCB$B_STATE                   00000052    D
UCB$B_TRACKS                  0000003D    D
UCB$B_TT_CRFILL               0000009D    D
UCB$B_TT_DECRF                000000A1    D
UCB$B_TT_DELFF                000000A2    D
UCB$B_TT_DESPEE               000000A0    D
UCB$B_TT_DETYPE               000000A4    D
UCB$B_TT_LFFILL               0000009E    D
UCB$B_TT_SPEED                0000009C    D
UCB$B_TYPE                    0000000A    D
UCB$B_VERTSZ                  0000003F    D
UCB$C_LENGTH                  00000074    D
UCB$C_MB_LENGTH               00000090    D
UCB$C_TT_LENGTH               000000BC    D
UCB$K_LENGTH                  00000074    D
UCB$K_MB_LENGTH               00000090    D
UCB$K_TT_LENGTH               000000BC    D
UCB$L_AMB                     00000054    D
UCB$L_ASTQBL                  00000010    D
UCB$L_ASTQFL                  0000000C    D
UCB$L_CPID                    0000005C    D
UCB$L_CRB                     00000020    D
UCB$L_DDB                     00000024    D
UCB$L_DEVCHAR                 00000034    D
UCB$L_DEVDEPEND               0000003C    D
UCB$L_DPC                     00000080    D
UCB$L_DUETIM                  0000005C    D
UCB$L_DX_BFPNT                0000009C    D
UCB$L_DX_BUF                  00000098    D
UCB$L_DX_RXDB                 000000A0    D
UCB$L_EMB                     00000078    D
UCB$L_FIRST                   00000014    D
UCB$L_FPC                     0000000C    D
UCB$L_FQBL                    00000004    D
UCB$L_FQFL                    00000000    D
UCB$L_FR3                     00000010    D
UCB$L_FR4                     00000014    D
UCB$L_IOQBL                   00000044    D
UCB$L_IOQFL                   00000040    D
```

```
UCB$L_IRP                       0000004C    D
UCB$L_LINK                      0000002C    D
UCB$L_LOGADR                    0000J064    D
UCB$L_MAXBLOCK                  00000084    D
UCB$L_MB_MBX                    0000007C    D
UCB$L_MB_PORT                   0000008C    D
UCB$L_MB_RAST                   00000078    D
UCB$L_MB_SHB                    00000080    D
UCB$L_MB_WAST                   00000074    D
UCB$L_MB_WIOQBL                 00000088    D
UCB$L_MB_WIOQFL                 00000084    D
UCB$L_MEDIA                     0000008C    D
UCB$L_NT_DATSSB                 00000074    D
UCB$L_NT_INTSSB                 00000078    D
UCB$L_OPCNT                     00000060    D
UCB$L_OWNUIC                    0000001C    D
UCB$L_PID                       00000028    D
UCB$L_RQBL                      00000004    D
UCB$L_RQFL                      00000000    D
UCB$L_SVAPTE                    00000068    D
UCB$L_SVPN                      00000064    D
UCB$L_TT_DECHAR                 000000A8    D
UCB$L_TT_RDUE                   0000008C    D
UCB$L_TT_RTIMOU                 000000B8    D
UCB$L_VCB                       00000030    D
UCB$M_INT                   =   00000002    D
UCB$M_TIM                   =   00000001    D
UCB$M_TIMOUT                =   00000040    D
UCB$T_PARTNER                   0000000C    D
UCB$V_TIM                   =   00000000    D
UCB$W_BCNT                      0000006E    D
UCB$W_BCR                       00000096    D
UCB$W_BOFF                      0000006C    D
UCB$W_BUFQUO                    00000018    D
UCB$W_BYTESTOGO                 0000003E    D
UCB$W_CHARGE                    0000004A    D
UCB$W_CYLINDERS                 0000003E    D
UCB$W_DA                        0000008C    D
UCB$W_DC                        0000008E    D
UCB$W_DEVBUFSIZ                 0000003A    D
UCB$W_DEVSTS                    0000005A    D
UCB$W_DIRSEQ                    00000088    D
UCB$W_DSTADDR                   00000018    D
UCB$W_DX_BCR                    000000A4    D
UCB$W_EC1                       00000090    D
UCB$W_EC2                       00000092    D
UCB$W_ERRCNT                    00000072    D
UCB$W_FUNC                      0000007E    D
UCB$W_MB_SEED                   00000000    D
UCB$W_MSGCNT                    00000016    D
UCB$W_MSGMAX                    00000014    D
UCB$W_NT_CHAN                   0000007C    D
UCB$W_OFFSET                    0000008A    D
UCB$W_REFC                      00000050    D
UCB$W_SIZE                      00000008    D
UCB$W_SRCADDR                   0000001A    D
UCB$W_STS                       00000058    D
UCB$W_TT_DESIZE                 000000A5    D
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000001A    D
WAITMS$_NARGS               =   00000002    D
WAITMS$_TAG                 =   00000008    D
WAITMS$_TIME                =   00000004    D
WAITMS_ARGS                     0000004C R  D  02
WAITUS$_NARGS               =   00000002    D
WAITUS$_TAG                 =   00000008    D
WAITUS$_TIME                =   00000004    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000068 (104.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000006 (6.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000426 (1062.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000004	309 (1)	243 (1) 244 (1) 254 (1) 257 (1) 258 (1) 309 (1)
$$T1	=00000000	353 (1)	243 (1) 244 (1) 254 (1) 257 (1) 258 (1) 309 (1) 353 (1)
$ER	=00000002	964 (1)	#-964 (1)
$MODULE	00000000-R	318 (1)	964 (1)
BIT...	=0000001A	260 (1)	247 (1) 259 (1) 260 (1)
CANTIM$_ACMODE	=00000008	243 (1)	
CANTIM$_NARGS	=00000002	243 (1)	
CANTIM$_REQIDT	=00000004	243 (1)	#-668 (1)
CLK$RE_INIT	000003D8-R	1077 (1)	#-563 (1) #-622 (1)
CLK$RE_SYNC_X	00000425-R	1110 (1)	#-1099 (1)
CLOCK_IPL	=00000018	276 (1)	
CMK$CANTIM	0000017C-R	665 (1)	#-658 (1)
CMK$HIBER	00000151-R	616 (1)	#-607 (1)
CMK$HIBER_X	00000166-R	624 (1)	#-618 (1)
CMK$SETIMR	00000215-R	821 (1)	#-815 (1)
CMK$_	=00000004	259 (1)	
CMK$_ASTEXIT	=00000000	259 (1)	
CMK$_CANTIM	=0000000B	259 (1)	#-658 (1)
CMK$_HIBER	=00000007	259 (1)	#-607 (1)
CMK$_INITSCB	=00000008	259 (1)	
CMK$_LAST	=0000000E	259 (1)	
CMK$_MMENABLE	=00000003	259 (1)	
CMK$_RCONSOLE	=00000001	259 (1)	
CMK$_REFRESH	=00000005	259 (1)	
CMK$_SCHDWK	=00000006	259 (1)	
CMK$_SETIMR	=0000000C	259 (1)	#-815 (1)
CMK$_SETIPL	=00000009	259 (1)	
CMK$_SETPRT	=0000000D	259 (1)	
CMK$_SGIPR	=0000000A	259 (1)	
CMK$_TCONSOLE	=00000002	259 (1)	
DDB$L_LINK	00000000		1013 (1) #-1014 (1)
DDB$L_UCB	00000004		1017 (1)
DS$CANWAIT	00000000-XR		309 (1)
DS$GA_SOFTINT	00000000-XR		918 (1)
DS$GB_STOPPING_THE_TIMER	00000000-XR		#-892 (1)
DS$GK_USOVR	00000000-XR		#-530 (1)
DS$GL_FLAGS	00000000-XR		#-393 (1) 524 (1) #-527 (1) 546 (1) 606 (1) 618 (1) 824 (1) #-831 (1)
DS$GL_TIMQFL	00000000-R	284 (1)	285 (1) 286 (1) 667 (1) 720 (1) 758 (1) #-908 (1) #-909 (1) #-946 (1) #-947 (1)
DS$GQ_CURYEAR	00000008-R	288 (1)	#-1105 (1) #-1106 (1)
DS$GT_NEWYEAR	00000010-R	291 (1)	294 (1)
DS$K_ERROR	=00000002	247 (1)	
DS$K_NORMAL	=00000001	247 (1)	
DS$K_NYSIZ	=00000014	294 (1)	
DS$K_SEVERE	=00000004	247 (1)	
DS$K_SUBSYS	=00000066	247 (1)	247 (1)

```
DS$K_WARNING                    =00000000       247     (1)
DS$M_ABRTFLG                    =00000040       260     (1)     #-392    (1)     #-526    (1)     #-830    (1)
DS$M_BADTIME                    =00100000       260     (1)
DS$M_BATCH                      =00400000       260     (1)
DS$M_BRKCLR                     =00001000       260     (1)
DS$M_BRKPT                      =00000800       260     (1)
DS$M_CHARFLG                    =00000100       260     (1)
DS$M_CMDFLG                     =00000080       260     (1)
DS$M_CTRLC                      =00000001       260     (1)
DS$M_CTRLO                      =00010000       260     (1)
DS$M_DEVFLG                     =00000200       260     (1)
DS$M_DISABLCC                   =01000000       260     (1)
DS$M_DONFLG                     =00002000       260     (1)
DS$M_ERRFLG                     =00000010       260     (1)
DS$M_EXCEPT                     =00080000       260     (1)
DS$M_EXETST                     =00040000       260     (1)
DS$M_HLTFLG                     =00000008       260     (1)
DS$M_LODFLG                     =00000002       260     (1)
DS$M_MEMMGT                     =00008000       260     (1)
DS$M_OUTPUT                     =00800000       260     (1)
DS$M_RUBFLG                     =00000020       260     (1)
DS$M_SCRIPT                     =00200000       260     (1)
DS$M_SETIMR                     =02000000       260     (1)
DS$M_STRFLG                     =00000004       260     (1)
DS$M_SUBT                       =00004000       260     (1)
DS$M_SYSFLG                     =00000400       260     (1)
DS$M_TIMRON                     =00020000       260     (1)
DS$V_ABRTFLG                    =00000006       260     (1)     #-545    (1)     #-605    (1)
DS$V_BADTIME                    =00000014       260     (1)
DS$V_BATCH                      =00000016       260     (1)
DS$V_BRKCLR                     =0000000C       260     (1)
DS$V_BRKPT                      =0000000B       260     (1)
DS$V_CHARFLG                    =00000008       260     (1)
DS$V_CMDFLG                     =00000007       260     (1)
DS$V_CTRLC                      =00000000       260     (1)
DS$V_CTRLO                      =00000010       260     (1)
DS$V_DEVFLG                     =00000009       260     (1)
DS$V_DISABLCC                   =00000018       260     (1)
DS$V_DONFLG                     =0000000D       260     (1)
DS$V_ERRFLG                     =00000004       260     (1)
DS$V_EXCEPT                     =00000013       260     (1)
DS$V_EXETST                     =00000012       260     (1)
DS$V_HLTFLG                     =00000003       260     (1)
DS$V_LODFLG                     =00000001       260     (1)
DS$V_MEMMGT                     =0000000F       260     (1)
DS$V_OUTPUT                     =00000017       260     (1)
DS$V_RUBFLG                     =00000005       260     (1)
DS$V_SCRIPT                     =00000015       260     (1)
DS$V_SETIMR                     =00000019       260     (1)
DS$V_STRFLG                     =00000002       260     (1)
DS$V_SUBT                       =0000000E       260     (1)
DS$V_SYSFLG                     =0000000A       260     (1)
DS$V_TIMRON                     =00000011       260     (1)     #-523    (1)     #-617    (1)     #-824    (1)
DS$WAITUS                        00000000-XR                      448     (1)
DS$_ARITH                       =006600D0       247     (1)
DS$_ASBE                        =00660118       247     (1)
DS$_BADLINK                     =006600F0       247     (1)
```

```
DS$_BADTYPE                     =006600E3       247     (1)
DS$_BIIC                        =0066012C       247     (1)
DS$_CHME                        =006600A8       247     (1)
DS$_CHMK                        =006600E0       247     (1)
DS$_DEVNAME                     =00660108       247     (1)
DS$_ERROR                       =00660002       247     (1)
DS$_FHWE                        =00660068       247     (1)
DS$_FRAGBUF                     =00660080       247     (1)
DS$_ICBUSY                      =006600C8       247     (1)
DS$_ICERR                       =006600C0       247     (1)
DS$_IHWE                        =00660060       247     (1)
DS$_ILLCHAR                     =00660018       247     (1)
DS$_ILLPAGCNT                   =00660078       247     (1)
DS$_ILLUNIT                     =00660100       247     (1)
DS$_INSFMEM                     =00660C50       247     (1)
DS$_IPL2HI                      =006600B8       247     (1)     #-833     (1)
DS$_IVADDR                      =00660040       247     (1)
DS$_IVVECT                      =00660038       247     (1)
DS$_KRNLSTK                     =00660090       247     (1)
DS$_LOGIC                       =00660070       247     (1)
DS$_MCHK                        =00660088       247     (1)
DS$_MMOFF                       =00660058       247     (1)
DS$_NEEDUNIT                    =0066C0F8       247     (1)
DS$_NODE                        =00660128       247     (1)
DS$_NOPCS                       =00660110       247     (1)
DS$_NORMAL                      =00660001       247     (1)
DS$_NOSUPPORT                   =006600B1       247     (1)
DS$_NOTDON                      =00660030       247     (1)
DS$_NOTIMP                      =006600B0       247     (1)       247     (1)     #-838     (1)
DS$_NULLSTR                     =00660010       247     (1)
DS$_OVERFLOW                    =00660008       247     (1)
DS$_POWER                       =00660098       247     (1)
DS$_PROGERR                     =00660020       247     (1)     #-452     (1)     #-532     (1)
DS$_SEVERE                      =00660004       247     (1)
DS$_TRANSL                      =006600A0       247     (1)
DS$_TRUNCATE                    =00660028       247     (1)
DS$_UNEXPINT                    =006600D8       247     (1)
DS$_VASFULL                     =00660048       247     (1)
DS$_WARNING                     =00660000       247     (1)       247     (1)
DSI$TIMER                        000002E8-R     938     (1)
DSI$TIMER_X                      0000037D-R     1000    (1)
DSI$TIMSRV                       00000288-R     886     (1)
DSR$SETIMR_INI                   000001B4-R     719     (1)
DSX$CANWAIT                      00000000-R     351     (1)
DSX$WAITMS                       00000027-R     438     (1)
DSX$WAITMS                       000000AF-R     475     (1)     #-449     (1)     #-455     (1)     #-458     (1)
DSX$WAITMS                       000000B0-R     520     (1)
DSX$WAITMS_SUBR                  000000C8-R     528     (1)
DSX$WAITU...                     00000130-R     567     (1)     #-534     (1)
DS_ERRSUP                        00000000-XR                      964     (1)
DYN$C_TQE                       =0000000F                         299     (1)
EXE$ALLOCTQE                     00000000-XR                     #-844     (1)
EXE$CANTIM                       00000167-R     656     (1)
EXE$DEANONPAGED                  000000C0-XR                     #-687     (1)     #-981     (1)
EXE$GL_ABSTIM                    00000000-XR                     #-1010    (1)     #-1027    (1)
EXE$GQ_SYSTIME                   00000000-XR                     #-1107    (1)     #-723     (1)     #-854     (1)     #-889     (1)
                                                                 #-890     (1)     #-911     (1)     #-914     (1)     #-951     (1)
```

```
                                                #-954     (1)

EXE$HIBER                        00000131-R     602       (1)
EXE$HIBER_X                      00000150-R     609       (1)
EXE$INSTIMQ                      000001D1-R     754       (1)    #-725     (1)    #-857     (1)    #-994     (1)
EXE$SETIMR                       00000203-R     813       (1)
EXE$WAKE                         0000018B-R     390       (1)
GETTIM$_NARGS                   =00000001       244       (1)
GETTIM$_TIMADR                  =00000004       244       (1)
ICCS$M_ERR                      =80000000       266       (1)    #-1083    (1)      268     (1)    #-537     (1)    #-541     (1)
                                                                 #-549     (1)    #-619     (1)    #-905     (1)
ICCS$M_IE                       =00000040       264       (1)    #-1087    (1)      268     (1)
ICCS$M_INT                      =00000080       265       (1)    #-1084    (1)      268     (1)    #-536     (1)    #-541     (1)
                                                                 #-548     (1)    #-620     (1)    #-905     (1)
ICCS$M_RESET                    =800000C1       268       (1)    #-899     (1)
ICCS$M_RUN                      =00000C01       262       (1)    #-1089    (1)      268     (1)    #-540     (1)
ICCS$M_XFER                     =00000010       263       (1)    #-1088    (1)    #-540     (1)
IOC$GL_DEVLIST                   00000000-XR                     1013      (1)
IOC$K_DIAGPID                    00000000-XR                    #-848     (1)
IPL$_HWCLK                      =00000018                       #-945     (1)
IPL$_POWER                      =0000001F                       #-1025    (1)
IPL$_TIMER                      =00000007                       #-1039    (1)    #-757     (1)    #-917     (1)    #-918     (1)
                                                                 #-961     (1)    #-999     (1)
KB_CHECK                         00000000-XR                    #-604     (1)    #-823     (1)
MSTIME                           00000060-R     311       (1)      309     (1)    #-454     (1)
ONESECTIM                        00000024-R     296       (1)    #-1108    (1)      724     (1)
ONE_SECOND                       00000380-R     1009      (1)      301     (1)
ONLY_CLOCK_INIT                  00000000-XR                     1096      (1)
PR$_ICCS                        =00000018                       #-1085    (1)    #-1090    (1)    #-537     (1)    #-542     (1)
                                                                 #-547     (1)    #-620     (1)    #-895     (1)    #-906     (1)
PR$_ICR                         =0000001A                       #-553     (1)    #-621     (1)
PR$_IPL                         =00000012                       #-1025    (1)    #-1033    (1)    #-1039    (1)    #-521     (1)
                                                                 #-533     (1)    #-543     (1)    #-666     (1)    #-691     (1)
                                                                 #-757     (1)    #-769     (1)    #-945     (1)    #-961     (1)
                                                                 #-999     (1)
PR$_NICR                        =00000019                       #-1086    (1)    #-539     (1)
PR$_SIRR                        =00000014                       #-917     (1)
PSL$S_IPL                       =00000005                       #-443     (1)    #-835     (1)
PSL$V_IPL                       =00000010                       #-443     (1)    #-835     (1)
PSL$V_IS                        =0000001A       259       (1)    #-607     (1)    #-658     (1)    #-815     (1)
SCH$QAST                         00000000-XR                    #-977     (1)
SEC_TICK                        =00000064       274       (1)
SETIMR$_ASTADR                  =0000000C       309       (1)    #-849     (1)
SETIMR$_DAYTIM                  =00000008       309       (1)    #-839     (1)    #-853     (1)
SETIMR$_EFN                     =00000004       309       (1)    #-851     (1)
SETIMR$_NARGS                   =00000004       309       (1)
SETIMR$_REQIDT                  =00000010       309       (1)
SIZ...                          =00000001       260       (1)      260     (1)
SS$_EXQUOTA                     =0000001C                       #-522     (1)    #-822     (1)
SS$_NORMAL                      =00000001                       #-394     (1)    #-473     (1)    #-560     (1)    #-608     (1)
                                                                 #-659     (1)    #-858     (1)
SYS$CANTIM                       00000000-XR                      353     (1)
SYS$CLREF                        00000000-XR                      827     (1)
SYS$GETTIM                       00000000-XR                      440     (1)      462     (1)
SYS$HIBER                        00000000-XR                      460     (1)
SYS$SETEF                        00000000-XR                      972     (1)
SYS$SETIMR                       00000000-XR                      457     (1)
SYS$WAKE                         00000000-XR                      360     (1)
```

```
TICK_NS                         =000186A0     273     (1)     #-1103    (1)     #-453     (1)     #-889     (1)
TICK_US                         =00002710     272     (1)     #-1086    (1)       273     (1)       274     (1)
TODR_BIAS                       =10000000     270     (1)     #-1098    (1)
TQE$B_EFN                       =00000029                     #-852     (1)     #-971     (1)
TQE$B_RQTYPE                    =0000000B                     #-677     (1)     #-847     (1)       962     (1)       990     (1)
TQE$C_SSREPT                    =00000005                       300     (1)
TQE$L_AST                       =00000010                     #-850     (1)     #-973     (1)
TQE$L_ASTPRM                    =00000014                     #-681     (1)
TQE$L_FPC                       =0000000C                       989     (1)
TQE$L_FR3                       =00000010                     #-988     (1)
TQE$L_PID                       =0000000C                     #-848     (1)
TQE$L_TQBL                      =00000004                     #-760     (1)
TQE$L_TQFL                      =00000000                     #-675     (1)
TQE$Q_DELTA                     =00000020                     #-991     (1)
TQE$Q_TIME                      =00000018                     #-1108    (1)     #-756     (1)     #-763     (1)     #-766     (1)
                                                              #-911     (1)     #-914     (1)     #-951     (1)     #-954     (1)
                                                              #-992     (1)     #-993     (1)
TQE$V_REPEAT                    =00000002                     #-990     (1)
UCB$B_DIPL                       00000052                     #-1033    (1)
UCB$L_DUETIM                     0000005C                     #-1027    (1)
UCB$L_FPC                        0000000C                     #-1035    (1)
UCB$L_FR3                        00000010                     #-1034    (1)
UCB$L_LINK                       0000002C                       1017    (1)     #-1019    (1)
UCB$M_INT                       =00000002                     #-1030    (1)
UCB$M_TIM                       =00000001                     #-1030    (1)
UCB$M_TIMOUT                    =00000040                     #-1032    (1)
UCB$V_TIM                       =000C0000                     #-1026    (1)
UCB$W_STS                        00000058                       1026    (1)     #-1031    (1)     #-1032    (1)
WAITMS$_NARGS                   =00000002     257     (1)
WAITMS$_TAG                     =00000008     257     (1)     #-467     (1)
WAITMS$_TIME                    =00000004     257     (1)     #-446     (1)     #-453     (1)     #-469     (1)
WAITMS_ARGS                      0000004C-R   307     (1)       457     (1)
WAITUS$_NARGS                   =00000002     258     (1)
WAITUS$_TAG                     =00000008     258     (1)     #-556     (1)     #-558     (1)
WAITUS$_TIME                    =00000004     258     (1)     #-529     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANTIMDEF	1	243 (1)	243 (1)
$CANTIM_S	1	353 (1)	353 (1)
$CLREF_G	1	827 (1)	827 (1)
$DDBDEF	1	245 (1)	245 (1)
$DEF	1	260 (1)	
$DEFINI	1	245 (1)	245 (1) 246 (1) 248 (1) 249 (1) 250 (1)
			251 (1) 252 (1) 253 (1) 255 (1) 256 (1)
$DS_DSADEF	5	246 (1)	246 (1)
$DS_DSDEF	2	247 (1)	247 (1)
$DS_SCBDEF	3	251 (1)	251 (1)
$DS_WAITMS_DEF	1	257 (1)	257 (1)
$DS_WAITUS_DEF	1	258 (1)	258 (1)
$DS_WAITUS_S	1	448 (1)	448 (1)
$DYNDEF	2	248 (1)	248 (1)
$EQU	1	260 (1)	247 (1) 259 (1)
$EQULS1	1	259 (1)	247 (1) 259 (1)
$EQULST	1	247 (1)	247 (1) 259 (1)
$GBLINI	2		247 (1) 259 (1) 260 (1)
$GETTIMDEF	1	244 (1)	244 (1)
$GETTIM_S	1	440 (1)	440 (1) 462 (1)
$HIBER_S	1	460 (1)	460 (1)
$IPLDEF	1	249 (1)	249 (1)
$OFFDEF	1	243 (1)	243 (1) 244 (1) 254 (1) 257 (1) 258 (1)
			309 (1)
$PRDEF	4	152 (1)	250 (1)
$PSLDEF	2	253 (1)	253 (1)
$PUSHADR	1	440 (1)	440 (1) 448 (1) 462 (1) 964 (1)
$PUSHTWO	1	353 (1)	353 (1)
$SETIMR	1	308 (1)	308 (1)
$SETIMRDEF	1	254 (1)	254 (1) 309 (1)
$SETIMR_G	1	457 (1)	457 (1)
$SSDEF	21	252 (1)	252 (1)
$TQEDEF	2	255 (1)	255 (1)
$UCBDEF	10	256 (1)	256 (1)
$VIELD	1		260 (1)
$VIELD1	1	260 (1)	260 (1)
CASE	1	963 (1)	963 (1)
CMK	1	607 (1)	607 (1) 658 (1) 815 (1)
CMKDEF	1	259 (1)	259 (1)
DSBINT	1	521 (1)	521 (1) 666 (1) 757 (1)
DSFDEF	3	260 (1)	260 (1)
ENBINT	1	533 (1)	533 (1) 543 (1) 691 (1) 769 (1)
ERRSUP_S	1	964 (1)	964 (1)
MODNAM	1	318 (1)	318 (1)
SETIPL	1	945 (1)	1025 (1) 1033 (1) 1039 (1) 945 (1) 961 (1)
			999 (1)
SOFTINT	1	917 (1)	917 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.11	00:00:00.22
Command processing	142	00:00:00.73	00:00:01.56
Pass 1	1077	00:00:19.53	00:00:27.59
Symbol table sort	3	00:00:01.73	00:00:01.93
Pass 2	345	00:00:03.93	00:00:04.97
Symbol table output	51	00:00:00.35	00:00:00.42
Psect synopsis output	8	00:00:00.04	00:00:00.04
Cross-reference output	119	00:00:01.09	00:00:01.42
Assembler run totals	1784	00:00:27.53	00:00:38.15

The working set limit was 1000 pages.
106889 bytes (209 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1107 non-local and 42 local symbols.
1114 source lines were read in Pass 1, producing 0 object records in Pass 2.
141 pages of virtual memory were used to define 42 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	10
DRB1:[DS.WORK]DS.MLB;218	5
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	6
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	40

1329 GETS were required to define 40 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CLOCK/UPDA=(CLOCK.UPD,CLOCK.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT

Table of contents

(2) 73 COM$DELATTNAST - DELIVER ATTENTION ASTS
(3) 111 COM$FLUSHATTNS - FLUSH ATTENTION AST LIST
(4) 154 COM$POST - POST I/O COMPLETION INDEPENDENT OF UNIT STATUS
(5) 185 COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
(6) 230 COM$SETATTNAST - SET UP ATTENTION AST

```
02                      0000     .1          .TITLE  COMDRVSUB       *** COMDRVSUB driver support routines
02                      0000     .2          .IDENT  /06.01/
-2                      0000      3 ;
                        0000      4 ;*****************************************************************************
                        0000      5 ;*                                                                           *
                        0000      6 ;*  COPYRIGHT (c) 1978, 1979, 1980                                           *
                        0000      7 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      8 ;*                                                                           *
                        0000      9 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000     10 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     11 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000     12 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000     13 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000     14 ;*  TRANSFERRED.                                                            *
                        0000     15 ;*                                                                           *
                        0000     16 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000     17 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                        0000     18 ;*  CORPORATION.                                                            *
                        0000     19 ;*                                                                           *
                        0000     20 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000     21 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     22 ;*                                                                           *
                        0000     23 ;*****************************************************************************
                        0000     24 ;
                        0000     25 ;++
                        0000     26 ; FACILITY:
                        0000     27 ;
                        0000     28 ;       VAX/VMS I/O DRIVERS
                        0000     29 ;
                        0000     30 ; ABSTRACT:
                        0000     31 ;
                        0000     32 ;       THIS MODULE CONTAINS SUBROUTINES FOR THE TERMINAL,MAILBOX AND DMC11 DRIVERS.
                        0000     33 ;
                        0000     34 ; AUTHOR:
                        0000     35 ;
                        0000     36 ;       R.HEINEN 8-SEPT-1977
                        0000     37 ;
                        0000     38 ; REVISION HISTORY:
                        0000     39 ;
                        0000     40 ;       V07     TCM0001         Trudy C Matthews        2-Jan-1980
                        0000     41 ;               Changed COM$SETATTNAST so that it only extracts the
                        0000     42 ;               two bits of the access mode in the P3 parameter,
                        0000     43 ;               instead of taking the whole byte.
                        0000     44 ;
02                      0000     45 ;
02                      0000     46 ;               Dave Butenhof
02                      0000     47 ;       01      Update VMS module for Supervisor; delete quota checks
                        0000     48 ;--
                        0000     49 ;
                        0000     50 ; EXTERNAL SYMBOLS
                        0000     51 ;
                        0000     52 ;       $ACBDEF                         ; DEFINE AST CONTROL BLOCK
                        0000     53 ;       $CCBDEF                         ; DEFINE CCB
                        0000     54 ;       $DYNDEF                         ; DEFINE DYNAMIC MEMORY BLOCKS
                        0000     55 ;       $IPLDEF                         ; DEFINE IPL LEVELS
                        0000     56 ;       $IRPDEF                         ; DEFINE I/O PACKET
                        0000     57 ;       $PCBDEF                         ; DEFINE PCB
```

```
                        0000     55          $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                        0000     56          $PRIDEF                         ; DEFINE NEW PRIORITIES
                        0000     57          $PRVDEF                         ; DEFINE PRIVELEGE BITS
                        0000     58          $PSLDEF                         ; DEFINE PSL
                        0000     59          $RSNDEF                         ; DEFINE RESOURCES
                        0000     60          $UCBDEF                         ; DEFINE UCB
                        0000     61 ;
                        0000     62 ; LOCAL DEFINITIONS
                        0000     63 ;
                00000000 0000    64 P1= 0
                00000004 0000    65 P2= 4
                00000008 0000    66 P3= 8
                0000000C 0000    67 P4= 12
                00000010 0000    68 P5= 16
                00000014 0000    69 P6= 20
                         0000    70
02                  0000C000     71          .PSECT  SEP, SHR,EXE,RD,WRT,LONG
```

```
                                0000     73          .SBTTL  COM$DELATTNAST - DELIVER ATTENTION ASTS
                                0000     74 ;++
                                0000     75 ; COM$DELATTNAST - DELIVER ATTENION ASTS
                                0000     76 ;
                                0000     77 ; FUNCTIONAL DESCRIPTION:
                                0000     78 ;
                                0000     79 ; THIS ROUTINE IS USED BY THE TERMINAL AND MAILBOX DRIVERS TO DELIVER
                                0000     80 ; ALL OF THE ASTS AWAITING ATTENTION. THE CONTROL BLOCKS ARE USED AS FORK BLOCKS
                                0000     81 ; TO IPL$_QUEUEAST.
                                0000     82 ;
                                0000     83 ; INPUTS:
                                0000     84 ;
                                0000     85 ;       R4 = ADDRESS OF LIST HEAD OF AST CONTROL BLOCKS
                                0000     86 ;       R5 = UCB OF UNIT
                                0000     87 ;
                                0000     88 ; OUTPUTS:
                                0000     89 ;
                                0000     90 ;       R2,R3,R4,R5 ARE PRESERVED.
                                0000     91 ;--
                                0000     92 COM$DELATTNAST::                             ; DELIVER ATTENTION ASTS
                          38    BB 0000  93          PUSHR   #^M<R3,R4,R5>              ;
                    55    04  BE  D0 0002  94 10$:    MOVL    @4(SP),R5                 ; GET NEXT ENTRY
                          28    13 0006  95          BEQL    50$                       ; IF EQL THEN NONE
                    04 BE     65  D0 0008  96          MOVL    (R5),@4(SP)               ; CLOSE LIST
                       F3     1F  9F 000C  97          PUSHAB  B^10$                     ; SET UP RETURN ADDRESS
                                  000F  98          FORK                              ; CREATE FORK PROCESS
                                  0015  99 ;
                                  0015 100 ; AST QUEUE FORK PROCESS
                                  0015 101 ;
                10 A5     18 A5  7D 0015 102          MOVQ    ACB$L_KAST(R5),ACB$L_AST(R5); REARRANGE ENTRIES
                08 A5     20 A5  90 001A 103          MOVB    ACB$L_KAST+8(R5),ACB$B_RMOD(R5);
                0C A5     24 A5  D0 001F 104          MOVL    ACB$L_KAST+12(R5),ACB$L_PID(R5);
                          18 A5  D4 0024 105          CLRL    ACB$L_KAST(R5)            ;
                       52    01  9A 0027 106          MOVZBL  #PRI$_IOCOM,R2            ; SET UP PRIORITY INCREMENT
                00000000'GF     17 002A 107          JMP     G^SCH$QAST                ; QUEUE THE AST
                          38    BA 0030 108 50$:     POPR    #^M<R3,R4,R5>             ;
                                05 0032 109          RSB
```

```
                                0033   111          .SBTTL  COM$FLUSHATTNS - FLUSH ATTENTION AST LIST
                                0033   112 ;++
                                0033   113 ; COM$FLUSHATTNS - FLUNS ATTENTION AST LIST
                                0033   114 ;
                                0033   115 ; THIS ROUTINE IS USED BY THE TERMINAL AND MAILBOX DRIVERS TO FLUSH
                                0033   116 ; AN ATTENTION AST LIST. THIS IS DONE AT CANCEL I/O TIME AND WHEN A
                                0033   117 ; QIO SPECIFIES A 0 AST ADDRESS ON A SET ATTENTION AST FUNCTION.
                                0033   118 ; IF THE AST CONTROL BLOCK OWNER IS NO LONGER IN THE SYSTEM THE AST IS ALSO
                                0033   119 ; FLUSHED.
                                0033   120 ;
                                0033   121 ;
                                0033   122 ; INPUTS:
                                0033   123 ;
                                0033   124 ;       R4 = PCB ADDRESS
                                0033   125 ;       R5 = UCB ADDRESS OF RELATED UNIT
                                0033   126 ;       R6 = CHANNEL NUMBER
                                0033   127 ;       R7 = LIST HEAD
                                0033   128 ;
                                0033   129 ; OUTPUTS:
                                0033   130 ;
                                0033   131 ;       R0 = SS$_NORMAL
                                0033   132 ;       R1,R2,R7 ARE DISTROYED.
                                0033   133 ;
                                0033   134 ;--
                                0033   135 COM$FLUSHATTNS::                          ; FLUSH ATTENTION AST LIST
                                0033   136          DSBINT  UCB$B_DIPL(R5)         ; DISABLE INTERRUPTS
                     50  67    D0  003A   137 10$:  MOVL    (R7),R0                ; GET LIST ENTRY
                         13    13  003D   138       BEQL    50$                    ; IF EQL THEN DONE
                24 A0    60 A4   D1  003F   139       CMPL    PCB$L_PID(R4),ACB$L_KAST+12(R0); PID MATCH?
                         10    12  0044   140       BNEQ    40$                    ; IF NEQ THEN NO
                   22 A0   56    B1  0046   141       CMPW    R6,ACB$L_KAST+10(R0)   ; CHANNEL MATCH?
                         0A    12  004A   142       BNEQ    40$                    ; IF NEQ THEN NO
                      67   60    D0  004C   143       MOVL    (R0),(R7)              ; CLOSE UP LIST TO REMOVE ENTRY
                                   004F   144       ENBINT                         ; REENABLE INTERRUPTS
-1                       1F    10  0052   145       BSBB    COM$DRVDEALMEM         ; DEALLOCATE THE BLOCK
                         DD    11  0054   146       BRB     COM$FLUSHATTNS         ; CONTINUE
                      57   50    D0  0056   147 40$:  MOVL    R0,R7                  ; LOOK TO NEXT ENTRY
                         DF    11  0059   148       BRB     10$                    ; CONTINUE
                                   005B   149 50$:  ENBINT                         ; REENABLE INTERRUPTS
                     50   00'8F  9A  005E   150       MOVZBL  #SS$_NORMAL,R0         ; SET NORMAL RETURN
                                05  0062   151       RSB
```

```
                                0063   154          .SBTTL  COM$POST - POST I.O COMPLETION INDEPENDENT OF UNIT STATUS
                                0063   155 ;++
                                0063   156 ; COM$POST - POST I/O COMPLETION INDEPENDENT OF UNIT STATUS
                                0063   157 ;
                                0063   158 ; FUNCTIONAL DESCRIPTION:
                                0063   159 ;
                                0063   160 ; THIS ROUTINE IS USED BY THE TERMINAL, MAILBOX AND DMC DRIVER TO COMPLETE
                                0063   161 ; I/O OPERATIONS INDEPENDENT OF THE STATUS OF THE UNIT. NO ATTEMPT IS MADE
                                0063   162 ; TO DE-QUEUE ANOTHER PACKET OR CHANGE THE BUSY STATUS OF THE UNIT.
                                0063   163 ;
                                0063   164 ; INPUTS:
                                0063   165 ;
                                0063   166 ;       R3 = I/O PACKET ADDRESS
                                0063   167 ;       R5 = UCB ADDRESS
                                0063   168 ;
                                0063   169 ; IMPLICIT INPUTS:
                                0063   170 ;
                                0063   171 ;       CALLER AT DRIVER FORK IPL OR GREATER.
                                0063   172 ;       IRP$L_MEDIA AND IRP$L_MEDIA+4 ARE THE IOSB QUAD WORD.
                                0063   173 ;
                                0063   174 ; OUTPUTS:
                                0063   175 ;
                                0063   176 ;       R0,R1 ARE DISTROYED.
                                0063   177 ;--
                                0063   178 COM$POST::                                ; COMPLETE I/O
                      6C A5   D6 0063   179          INCL    UCB$L_OPCNT(R5)         ; INCREMENT OPERATION COUNT
                         63   0E 0066   180          INSQUE  (R3),@IOC$GL_PSBL       ; INSERT PACKET ON QUEUE
                00000000'FF      0068
                         03   12 006D   181          BNEQ    10$                     ; IF NEQ THEN NOT FIRST
                                 006F   182          SOFTINT #IPL$_IOPOST            ; REQUEST FORK
                                 05 0072   183 10$:    RSB                             ; RETURN
```

```
                                0073   185          .SBTTL  COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
                                0073   186 ;++
                                0073   187 ; COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
                                0073   188 ;
                                0073   189 ; FUNCTIONAL DESCRIPTION:
                                0073   190 ;
                                0073   191 ; THIS ROUTINE IS USED BY DRIVERS TO DEALLOCATE SYSTEM DYNAMIC MEMORY.
                                0073   192 ;
                                0073   193 ; IT CAN BE CALLED AT ANY IPL.
                                0073   194 ;
                                0073   195 ; INPUTS:
                                0073   196 ;
                                0073   197 ;       R0 = ADDRESS OF THE BLOCK TO DEALLOCATE
                                0073   198 ;
                                0073   199 ;*********************************************************
                                0073   200 ;
                                0073   201 ; THE BUFFER MUST BE AT LEAST 24 BYTES LONG!
                                0073   202 ;
                                0073   203 ;*********************************************************
                                0073   204 ;
                                0073   205 ; OUTPUTS:
                                0073   206 ;
                                0073   207 ;       R0-R5 ARE PRESERVED.
                                0073   208 ;--
                                0073   209 COM$DRVDEALMEM::                          ; DEALLOCATE DRIVER MEMORY
                    08 A0   18  B1  0073   210          CMPW    #24,8(R0)           ; BIG ENOUGH BUFFER TO DEALLOCATE?
                            1E  1A  0077   211          BGTRU   30$                 ; IF GTRU THEN NO - ERROR
                            38  BB  0079   212          PUSHR   #^M<R3,R4,R5>       ; SAVE FORKING REGS
                     0B A0      06  90  007B   213          MOVB    #IPL$_QUEUEAST,11(R0) ; INSERT PROPER IPL
                        55      50  D0  007F   214          MOVL    R0,R5               ; COPY ADDRESS
                          94'AF     9F  0082   215          PUSHAB  B^20$               ; SET UP RETURN ADDRESS
                00000000'GF     16  C0  0085   216          JSB     G^EXE$FORK          ; CREATE FORK
                                    0088   217 ;
                                    0088   218 ; IPL$_QUEUEAST FORK ROUTINE
                                    0088   219 ;
                            50      55  D0  0088   220          MOVL    R5,R0               ; DEALLOCATE THE BLOCK
                00000000'GF     17  008E   221          JMP     G^EXE$DEANONPAGED   ;
                            38      BA  0094   222 20$:     POPR    #^M<R3,R4,R5>       ;
                                    05  0096   223          RSB                         ;
                                    0097   224 ;
                                    0097   225 ; BUGCHECK ON TOO SMALL A BUFFER
                                    0097   226 ;
02                                  0097     .1 30$:;   BUG_CHECK BADDALRQSZ        ; BUGCHECK
-1                                  05  0097   228          RSB                         ; CONTINUE
```

```
                     0098   230          .SBTTL  COM$SETATTNAST - SET UP ATTENTION AST
                     0098   231 ;++
                     0098   232 ; COM$SETATTNAST - SET UP ATTENTION AST
                     0098   233 ;
                     0098   234 ; FUNCTIONAL DESCRIPTION:
                     0098   235 ;
                     0098   236 ; THIS ROUTINE IS A SUBROUTINE USED BY THE TERMINAL AND MAILBOX DRIVERS
                     0098   237 ; TO PROCESS REQUESTS FOR ENABLE OR DISABLE OF ATTENTION ASTS.
                     0098   238 ; P1 IS THE ADDRESS OF THE AST SERVICE FOR ENABLES. P1 = 0 FOR DISABLE.
                     0098   239 ; FOR DISABLES, THE SPECIFIED LIST IS SEARCHED AND THE ENTRY EXTRACTED AND
                     0098   240 ; DEALLOCATED.
                     0098   241 ; FOR ENABLES, A CONTROL BLOCK IS SET UP THAT WILL DOUBLE AS THE AST CONTROL
                     0098   242 ; BLOCK WHEN THE AST IS DELIVERED. THE BLOCK IS FORMATTED AS FOLLOWS:
                     0098   243 ;
                     0098   244 ;       ACB$B_RMOD = IPR$_QUEUEAST
                     0098   245 ;       ACB$L_KAST = AST PC
                     0098   246 ;       ACB$L_KAST+4 = AST PARAMETER (P2)
                     0098   247 ;       ACB$L_KAST+8 = ACCESS MODE OF REQUEST
                     0098   248 ;       ACB$L_KAST+10 = CHANNEL NUMBER
                     0098   249 ;       ACB$L_KAST+12 = PID OF REQUEST
                     0098   250 ;
                     0098   251 ; THE NEW BLOCK IS PLACED AT THE HEAD OF THE CURRENT LIST.
                     0098   252 ;
                     0098   253 ; IN BOTH CASES THE I/O IS COMPLETED.
                     0098   254 ;
                     0098   255 ; INPUTS:
                     0098   256 ;
                     0098   257 ;       R3 = I/O PACKET ADDRESS
                     0098   258 ;       R4 = CURRENT PCB
                     0098   259 ;       R5 = UCB ADDRESS
                     0098   260 ;       R6 = ASSIGNED CCB
                     0098   261 ;       R7 = ADDRESS OF THE CONTROL AST LIST HEAD TO CHANGE
                     0098   262 ;       AP = ADDRESS OF THE QIO ARGLIST
                     0098   263 ;
                     0098   264 ; OUTPUTS:
                     0098   265 ;
                     0098   266 ;       R0 = STATUS OF THE I/O
                     0098   267 ;       R3 = PACKET ADDRESS
                     0098   268 ;       R5 = UCB ADDRESS
                     0098   269 ;
                     0098   270 ;       NO OTHER REGISTERS ARE PRESERVED.
                     0098   271 ;
                     0098   272 ; COMPLETION CODES:
                     0098   273 ;
                     0098   274 ;       SS$_NORMAL
                     0098   275 ;       SS$_EXQUOTA  -- BUFFERED I/O OR AST QUOTA FAILURE
                     0098   276 ;       SS$_INSUFMEM -- DYNAMIC MEMORY FAILURE
                     0098   277 ;--
                     0098   278 COM$SETATTNAST::                          ; SET UP ATTENTION AST
56   28 A3   3C      0098   279          MOVZWL  IRP$W_CHAN(R3),R6        ; GET PACKET CHANNEL NUMBER
   58   6C   D0      009C   280          MOVL    P1(AP),R8                ; GET USER AST ADDRESS
        92   13      009F   281          BEQL    COM$FLUSHATTNS           ; IF EQL THEN DISABLE FUNCTION
                     00A1   282 ;
                     00A1   283 ; REQUEST TO ENABLE AST
                     00A1   284 ;
                     00A1   285 ; SET UP AST BLOCK
                     00A1   286 ;
```

```
-4                51   28   9A  00A1    291          MOVZBL  #ACB$L_KAST+16,R1        ; SET SIZE OF NEEDED BLOCK
                  53   DD  00A4    292          PUSHL   R3                          ; SAVE PACKET ADDRESS
          00000000'GF   16  00A6    293          JSB     G^EXE$ALLOCBUF              ; ALLOCATE THE BUFFERED BLOCK
                  53 8ED0  00AC    294          POPL    R3                          ; RESTORE PACKET ADDRESS
                 40 50   E9  00AF    295          BLBC    R0,20$                      ; IF LOW SET THEN ALLOCATED
             0B A2   06   90  00B2    296          MOVB    #IPL$_QUEUEAST,ACB$B_RMOD(R2); INSERT FORK IPL
             18 A2   58   D0  00B6    297          MOVL    R8,ACB$L_KAST(R2)           ; INSERT AST ROUTINE ADDRESS
          1C A2   04   AC   D0  00BA    298          MOVL    P2(AP),ACB$L_KAST+4(R2)     ; INSERT PARAMETER FOR AST
                02   00   EF  00BF    299          EXTZV   #0,#2,P3(AP),R0             ; GET REQUEST ACCESS MODE
             50   08   AC        00C2
          00000000'GF   16  00C5    300          JSB     G^EXE$MAXACMODE             ; MAXIMIZE ACCESS MODE
             20 A2   50   9A  00CB    301          MOVZBL  R0,ACB$L_KAST+8(R2)         ; INSERT IN CONTROL BLOCK
          20 A2   40 8F   88  00CF    302          BISB    #ACB$M_QUOTA,ACB$L_KAST+8(R2); INSERT TARGET ACCESS MODE
             22 A2   56   B0  00D4    303          MOVW    R6,ACB$L_KAST+10(R2)        ; SAVE CHANNEL
          24 A2   60   A4   D0  00D8    304          MOVL    PCB$L_PID(R4),ACB$L_KAST+12(R2); INSERT PID ADDRESS OF REQUESTOR
                           00DD    305 ;
                           00DD    306 ; LOCK OUT INTERRUPTS TO ENTER BLOCK ON UCB
                           00DD    307 ;
                           00DD    308          DSBINT  UCB$B_DIPL(R5)              ; LOCK OUT INTERRUPTS
                62   67   D0  00E4    309          MOVL    (R7),(R2)                   ;  MERGE WITH CURRENT ENTRY
                67   52   D0  00E7    310          MOVL    R2,(R7)                     ; INSERT NEW ENTRY VALUE
                           00EA    311          ENBINT                              ; LOWER IPL
             50   00'8F   9A  00ED    312          MOVZBL  #SS$_NORMAL,R0              ; SET NORMAL RETURN
                     05   00F1    313          RSB                                 ; RETURN VIA CALLER
                   FF0B'  31  00F2    314 20$:     BRW     EXE$ABORTIO                 ; ABORT THE I/O
                           00F5    315 ;
                           00F5    316          .END
```

```
ACB$B_RMOD          0000000B    D          IRP$L_WIND          00000018    D          PCB$W_MTXCNT        0000000E    D
ACB$B_TYPE          0000000A    D          IRP$Q_NI_PRVMSK     0000003C    D          PCB$W_PPGCNT        00000036    D
ACB$C_LENGTH        00000018    D          IRP$W_ABCNT         0000003C    D          PCB$W_PRCCNT        00000042    D
ACB$K_LENGTH        00000018    D          IRP$W_BCNT          00000032    D          PCB$W_SIZE          00000008    D
ACB$L_AST           00000010    D          IRP$W_BOFF          00000030    D          PCB$W_STATE         0000002C    D
ACB$L_ASTPRM        00000014    D          IRP$W_CHAN          00000028    D          PCB$W_TMBU          00000032    D
ACB$L_ASTQBL        00000004    D          IRP$W_FUNC          00000020    D          PR$_IPL         =   00000012    D
ACB$L_ASTQFL        00000000    D          IRP$W_OBCNT         0000003E    D          PR$_SIRR        =   00000014    D
ACB$L_KAST          00000018    D          IRP$W_SIZE          00000008    D          PRI$_IOCOM      =   00000001    D
ACB$L_PID           0000000C    D          IRP$W_STS           0000002A    D          SCH$QAST            ********  X   02
ACB$M_QUOTA     =   00000040    D          IRP$W_TT_PRMPT      0000003C    D          SIZ...          =   00000002    D
ACB$W_SIZE          00000008    D          P1              =   00000000    D          SS$_NORMAL          ********  X   02
CCB$B_AMOD          00000009    D          P2              =   00000004    D          UCB$B_AMOD          00000053    D
CCB$B_STS           00000008    D          P3              =   00000008    D          UCB$B_CEX           00000077    D
CCB$C_LENGTH        00000010    D          P4              =   0000000C    D          UCB$B_CM1           0000004A    D
CCB$K_LENGTH        00000010    D          P5              =   00000010    D          UCB$B_CM2           0000004B    D
CCB$L_DIRP          0000000C    D          P6              =   00000014    D          UCB$B_DEVCLASS      00000038    D
CCB$L_UCB           00000000    D          PCB$B_ASTACT        0000000C    D          UCB$B_DEVTYPE       00000039    D
CCB$L_WIND          00000004    D          PCB$B_ASTEN         0000000D    D          UCB$B_DIPL          00000052    D
CCB$W_IOC           0000000A    D          PCB$B_PRI           0000000B    D          UCB$B_DX_SCTCNT     000000A6    D
COM$DELATTNAST      00000000 RG D  02      PCB$B_PRIB          0000002F    D          UCB$B_ERTCNT        00000070    D
COM$DRVDEALMEM      00000073 RG D  02      PCB$B_TYPE          0000000A    D          UCB$B_ERTMAX        00000071    D
COM$FLUSHATTNS      00000033 RG D  02      PCB$B_WEFC          0000002E    D          UCB$B_FEX           00000076    D
COM$POST            00000063 RG D  02      PCB$C_LENGTH        0000008C    D          UCB$B_FIPL          0000000B    D
COM$SETATTNAST      00000098 RG D  02      PCB$K_LENGTH        0000008C    D          UCB$B_LOCSRV        0000003C    D
EXE$ABORTIO         ********    X  02      PCB$L_ARB           00000084    D          UCB$B_OFFNDX        00000094    D
EXE$ALLOCBUF        ********    X  02      PCB$L_ASTQBL        00C00014    D          UCB$B_OFFRTC        00000095    D
EXE$DEANONF\GED     ********    X  02      PCB$L_ASTQFL        00000010    D          UCB$B_REMSRV        0000003D    D
EXE$FORK            ********    X  02      PCB$L_EFC2P         00000058    D          UCB$B_SECTORS       000C003C    D
EXE$MAXACMODE       ********    X  02      PCB$L_EFC3P         0000005C    D          UCB$B_SLAVE         00000074    D
IOC$GL_PSBL         ********    X  02      PCB$L_EFCS          00000050    D          UCB$B_SPR           00000075    D
IPL$_IOPOST     =   00000004    D          PCB$L_EFCU          00000054    D          UCB$B_STATE         00000052    D
IPL$_QUEUEAST   =   00000006    D          PCB$L_EFWM          0000004C    D          UCB$B_TRACKS        0000003D    D
IRP$B_CARCON        00000038    D          PCB$L_JIB           00000078    D          UCB$B_TT_CRFILL     0000009D    D
IRP$B_EFN           00000022    D          PCB$L_OWNER         0000001C    D          UCB$B_TT_DECRF      000000A1    D
IRP$B_PRI           00000023    D          PCB$L_PHD           00000064    D          UCB$B_TT_DELFF      000000A2    D
IRP$B_RMOD          0000000B    D          PCB$L_PHYPCB        00000018    D          UCB$B_TT_DESPEE     000000A0    D
IRP$B_TYPE          0000000A    D          PCB$L_PID           00000060    D          UCB$B_TT_DETYPE     000000A4    D
IRP$C_LENGTH        0000005C    D          PCB$L_PQB           0000004C    D          UCB$B_TT_LFFILL     0000009E    D
IRP$K_LENGTH        0000005C    D          PCB$L_SQBL          00000004    D          UCB$B_TT_SPEED      0000009C    D
IRP$L_ARB           00000050    D          PCB$L_SQFL          00000000    D          UCB$B_TYPE          0000000A    D
IRP$L_AST           00000010    D          PCB$L_STS           00000024    D          UCB$B_VERTSZ        0000003F    D
IRP$L_ASTPRM        00000014    D          PCB$L_UIC           00000088    D          UCB$C_LENGTH        00000074    D
IRP$L_DIAGBUF       00000044    D          PCB$L_WSSWP         00000020    D          UCB$C_MB_LENGTH     00000090    D
IRP$L_EXTEND        0000004C    D          PCB$L_WTIME         00000028    D          UCB$C_TT_LENGTH     000000BC    D
IRP$L_IOQBL         00000004    D          PCB$Q_PRIV          0000007C    D          UCB$K_LENGTH        0000C074    D
IRP$L_IOQFL         00000000    D          PCB$T_LNAME         00000068    D          UCB$K_MB_LENGTH     00000090    D
IRP$L_IOSB          00000024    D          PCB$T_TERMINAL      00000044    D          UCB$K_TT_LENGTH     000000BC    D
IRP$L_IOST1         00000034    D          PCB$W_APTCNT        00000030    D          UCB$L_AMB           00000054    D
IRP$L_IOST2         00000038    D          PCB$W_ASTCNT        00000038    D          UCB$L_ASTQBL        00000010    D
IRP$L_MEDIA         00000034    D          PCB$W_BIOCNT        0000003A    D          UCB$L_ASTQFL        0000000C    D
IRP$L_PID           0000000C    D          PCB$W_BIOLM         0000003C    D          UCB$L_CPID          0000005C    D
IRP$L_SEGVBN        00000040    D          PCB$W_DIOCNT        0000003E    D          UCB$L_CRB           00000020    D
IRP$L_SEQNUM        00000048    D          PCB$W_DIOLM         00000040    D          UCB$L_DDB           00000024    D
IRP$L_SVAPTE        0000002C    D          PCB$W_GPGCNT        00000034    D          UCB$L_DEVCHAR       00000034    D
IRP$L_TT_TERM       00000038    D          PCB$W_GRP           0000008A    D          UCB$L_DEVDEPEND     0000003C    D
IRP$L_UCB           0000001C    D          PCB$W_MEM           00000088    D          UCB$L_DPC           00000080    D
```

```
UCB$L_DUETIM     0000005C  D        UCB$W_MB_SEED    00000000  D
UCB$L_DX_BFPNT   0000009C  D        UCB$W_MSGCNT     00000016  D
UCB$L_DX_BUF     00000098  D        UCB$W_MSGMAX     00000014  D
UCB$L_DX_RXDB    000000A0  D        UCB$W_NT_CHAN    0000007C  D
UCB$L_EMB        00000078  D        UCB$W_OFFSET     0000008A  D
UCB$L_FIRST      00000014  D        UCB$W_REFC       00000050  D
UCB$L_FPC        0000000C  D        UCB$W_SIZE       00000008  D
UCB$L_FQBL       00000004  D        UCB$W_SRCADDR    0000001A  D
UCB$L_FQFL       00000000  D        UCB$W_STS        00000058  D
UCB$L_FR3        00000010  D        UCB$W_TT_DESIZE  000000A6  D
UCB$L_FR4        00000014  D        UCB$W_UNIT       00000048  D
UCB$L_IOQBL      00000044  D        UCB$W_VPROT      0000001A  D
UCB$L_IOQFL      00000040  D
UCB$L_IRP        0000004C  D
UCB$L_LINK       0000002C  D
UCB$L_LOGADR     00000064  D
UCB$L_MAXBLOCK   00000084  D
UCB$L_MB_MBX     0000007C  D
UCB$L_MB_PORT    0000008C  D
UCB$L_MB_RAST    00000078  D
UCB$L_MB_SHB     00000080  D
UCB$L_MB_WAST    00000074  D
UCB$L_MB_WIOQBL  00000088  D
UCB$L_MB_WIOQFL  00000084  D
UCB$L_MEDIA      0000008C  D
UCB$L_NT_DATSSB  00000074  D
UCB$L_NT_INTSSB  00000078  D
UCB$L_OFCNT      00000060  D
UCB$L_OWNUIC     0000001C  D
UCB$L_PID        00000028  D
UCB$L_RQBL       00000004  D
UCB$L_RQFL       00000000  D
UCB$L_SVAPTE     00000068  D
UCB$L_SVPN       00000064  D
UCB$L_TT_DECHAR  000000A8  D
UCB$L_TT_RDUE    000000CC  D
UCB$L_TT_RTIMOU  000000B8  D
UCB$L_VCB        0C000030  D
UCB$T_PARTNER    0000000C  D
UCB$W_BCNT       0000006E  D
UCB$W_BCR        00000096  D
UCB$W_BOFF       0000006C  D
UCB$W_BUFQUO     00000018  D
UCB$W_BYTESTOGO  0000003E  D
UCB$W_CHARGE     0000004A  D
UCB$W_CYLINDERS  0000003E  D
UCB$W_DA         0000008C  D
UCB$W_DC         0000008E  D
UCB$W_DEVBUFSIZ  0000003A  D
UCB$W_DEVSTS     0000005A  D
UCB$W_DIRSEQ     00000088  D
UCB$W_DSTADDR    00000018  D
UCB$W_DX_BCR     000000A4  D
UCB$W_ECT        00000090  D
UCB$W_EC2        00000092  D
UCB$W_ERRCNT     00000072  D
UCB$W_FUNC       0000007E  D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	000000F5 (245.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACB$B_RMOD	0000000B		#-103 (2) #-296 (6)
ACB$L_AST	00000010		#-102 (2)
ACB$L_KAST	00000018		#-102 (2) #-103 (2) #-104 (2) #-105 (2) #-139 (3) #-141 (3) #-291 (6) #-297 (6) #-298 (6) #-301 (6) #-302 (6) #-303 (6) #-304 (6)
ACB$L_PID	0000000C		#-104 (2)
ACB$M_QUOTA	=0000C040		#-302 (6)
COM$DELATTNAST	00000000-R	92 (2)	
COM$DRVDEALMEM	00000073-R	209 (5)	#-146 (3)
COM$FLUSHATTNS	00000033-R	135 (3)	#-147 (3) #-281 (6)
COM$POST	00000063-R	178 (4)	
COM$SETATTNAST	00000098-R	278 (6)	
EXE$ABORTIO	00000000-XR		#-314 (6)
EXE$ALLOCBUF	00000000-XR		293 (6)
EXE$DEANONPAGED	00000000-XR		221 (5)
EXE$FORK	00000000-XR		216 (5) 98 (2)
EXE$MAXACMODE	00000000-XR		300 (6)
IOC$GL_PSBL	00000000-XR		180 (4)
IPL$_IOPOST	=00000004		#-182 (4)
IPL$_QUEUEAST	=00000006		#-213 (5) #-296 (6)
IRP$W_CHAN	00000028		#-279 (6)
P1	=00000000	64 (1)	#-280 (6)
P2	=00000004	65 (1)	#-298 (6)
P3	=00000008	66 (1)	299 (6)
P4	=0000000C	67 (1)	
P5	=00000010	68 (1)	
P6	=00000014	69 (1)	
PCB$L_PID	00000060		#-139 (3) #-304 (6)
PR$_IPL	=00000012		#-136 (3) #-144 (3) #-150 (3) #-308 (6) #-311 (6)
PR$_SIRR	=00000014		#-182 (4)
PRI$_IOCOM	=00000001		#-106 (2)
SCH$QAST	00000000-XR		107 (2)
SS$_NORMAL	00000000-XR		#-151 (3) #-312 (6)
UCB$B_DIPL	00000052		#-136 (3) #-308 (6)
UCB$L_OPCNT	00000060		#-179 (4)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	49 (1)	49 (1)
$CCBDEF	1	50 (1)	50 (1)
$DEFINI	1	49 (1)	49 (1) 50 (1) 51 (1) 52 (1) 53 (1) 54 (1) 55 (1) 56 (1) 57 (1) 58 (1) 59 (1) 60 (1)
$DYNDEF	2	51 (1)	51 (1)
$IPLDEF	1	52 (1)	52 (1)
$IRPDEF	4	53 (1)	53 (1)
$PCBDEF	4	54 (1)	54 (1)
$PRDEF	4	55 (1)	55 (1)
$PRIDEF	1	56 (1)	56 (1)
$PRVDEF	4	57 (1)	57 (1)
$PSLDEF	2	58 (1)	58 (1)
$RSNDEF	1	59 (1)	59 (1)
$UCBDEF	10	60 (1)	60 (1)
DSBINT	1	136 (3)	136 (3) 308 (6)
ENBINT	1	144 (3)	144 (3) 150 (3) 311 (6)
FORK	1	98 (2)	98 (2)
SOFTINT	1	182 (4)	182 (4)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.11	00:00:00.22
Command processing	146	00:00:00.68	00:00:01.54
Pass 1	695	00:00:11.99	00:00:18.85
Symbol table sort	3	00:00:00.88	00:00:00.96
Pass 2	164	00:00:02.11	00:00:03.84
Symbol table output	21	00:00:00.15	00:00:00.30
Psect synopsis output	5	00:00:00.02	00:00:00.02
Cross-reference output	25	00:00:00.25	00:00:00.93
Assembler run totals	1096	00:00:16.22	00:00:26.68

The working set limit was 1000 pages.
52423 bytes (103 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 643 non-local and 9 local symbols.
314 source lines were read in Pass 1, producing 0 object records in Pass 2.
98 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYS0.SYSMAINT]DS.MLB;218	0
DMA1:[SYS0.SYSMAINT]DIAG.MLB;953	7
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	6
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	21

987 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) COMDRVSUB/UPDA=(COMDRVSUB.UPD,COMDRVSUB.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(3) 306 INI$LOAD_DEVICE Make ptables for console and boot devices
(4) 375 DSP$CONFIG_LOAD Configure load device
(9) 690 Common routines
(10) 804 configure RB730
(10) 930 Attach UDA-50/LESI load patn

```
0000      1          .TITLE  CONFIG  *** CONFIG Configure load device
0000      2          .IDENT  /06-19/
0000      3          .NoShow Conditionals
0000      4          .DSABL  GBL
0000      5
0000      6  ;
0000      7  ; Copyright (c) 1977, 1982, 1984
0000      8  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      9  ;
0000     10  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     11  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     12  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     13  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     14  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     15  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     16  ; REMAIN IN DEC.
0000     17  ;
0000     18  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     19  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     20  ; CORPORATION.
0000     21  ;
0000     22  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     23  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     24
0000     25  ;++
0000     26  ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     27  ;
0000     28  ; ABSTRACT:
0000     29  ;
0000     30  ; ENVIRONMENT:
0000     31  ;
0000     32  ; AUTHOR:       Roger Riggs     20-FEB-79       VERSION 01.
0000     33  ;
0000     34  ; MODIFICATIONS:
0000     35  ;
0000     36  ;                N. Howgate      26-Jun-79
0000     37  ;       01       References to I/O space are changed from
0000     38  ;                60000000 or 4000000 to IOC$K_IOSPACE
0000     39  ;                and are CPU dependant as defined in ONLYnnn.
0000     40  ;
0000     41  ;       02       John Ciukaj     1-25-80
0000     42  ;                Allow for RL01/RL02 to be configured: Add CON_UBA_RL
0000     43  ;
0000     44  ;       03       dave butenhof, 02-Sep-1981, version 6.5
0000     45  ;                Support RB730 (Nebula IDC) as load path.
0000     46  ;
0000     47  ;       04       - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     48  ;                Changed SEP Psect to CODE, DATA, and WORK.
0000     49  ;                Also added .LIBRARY statements for $DS,
0000     50  ;                $DIAG and SYS$LIBRARY:LIB.
0000     51  ;
0000     52  ;       05       - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     53  ;                Fixed truncation errors.
0000     54  ;
0000     55  ;       06       - Dave Butenhof, 24-Mar-1982, Version 6.6
0000     56  ;                Fix bug in CONFIG RB730 routine.  IDC can generate double
0000     57  ;                interrupt...make sure to disable device interrupts in handler
```

```
                                0000     58 ;               routine to avoid UNEXPECTED INTERRUPT error.
                                0000     59 ;       [07] Dave Butenhof, 15-Apr-1982, Version 6.7
                                0000     60 ;               Support UDA-50 load device.
                                0000     61 ;       [08] Dave Butenhof, 28-May-1982, version 6.8
                                0000     62 ;               Fix IDC code - use correct address for IDC, not the
                                0000     63 ;               one VMB gives us.
                                0000     64 ;
                                0000     65 ;       09      M. Baggett      24-Feb-1983     Version 6.11
                                0000     66 ;               Fixed document typo, support RC25.
                                0000     67 ;
                                0000     68 ;       10      M. Baggett      3-March-1983    Version 6.11
                                0000     69 ;               Added suport for HSC50.
                                0000     70 ;
                                0000     71 ;       11      M. Baggett      9-may-1983      Version 6.11
                                C000     72 ;               Booting from RP07 will have device name of DR and not DB.
                                0000     73 ;
                                0000     74 ;       12      M. Baggett      22-May-1983     Version 6.11
                                0000     75 ;               Allow any length name to be boot device (eg, CIAx, CIAxx, CIAxxx).
                                0000     76 ;
                                0000     77 ;       13      M. Baggett      21-Sep-1983     Version 6.13
                                0000     78 ;               Change HSC50 boot to a disk on the HSC50 to allow selections
                                0000     79 ;               among the different disks. Changed the name of LESI to RC11, and
                                0000     80 ;               temporarily changed the RC25 generic to DU.
                                0000     81 ;
                                0000     82 ;       14      M. Baggett      7-Sep-1983      Version 6.13
                                0000     83 ;               Change HSC50 controller naming to match standards.
                                0000     84 ;               Generic name is  HSC50$DUAnnn for DISK p-table.
                                0000     85 ;
                                0000     86 ;       15      Bob Bergazzi    2-14-84         Version 6.14
                                0000     87 ;               Moved code from KERNEL to here which builds ptables for the
                                0000     88 ;               console and boot devices. Subroutine is called INI$LOAD_DEVICE.
                                0000     89 ;               Done so that SET MEM can call this routine.
                                0000     90 ;
                                0000     91 ;       16      M. Baggett      15-Feb-1984     Version 6.14
                                0000     92 ;               Changed the default name for RC25/RCF25 device to DUA temporary ****
                                0000     93 ;
                                0000     94 ;       17      M. Baggett      16-May-1984     Version 7.0
                                0000     95 ;               Fixed HSC50 disk name to show proper unit number.
                                0000     96 ;
                                0000     97 ;       18      M. Baggett      6-Jun-1984      Version 7.0
                                0000     98 ;               Change device name for RC11/RC25/RCF25 back to DA.
                                0000     99 ;
                                0000    100 ;       19      Richard Brown   10-July-1984    Version 7.0
                                0000    101 ;               Change all references of "RC11" to "LESI".
                                0000    102 ;--
```

```
                        0000    104 ;                                                                                   [04]
                        0000    105 ; Include Files                                                                     [04]
                        0000    106 ;                                                                                   [04]
                        0000    107         .LIBRARY        /SYS$LIBRARY:LIB/       ;                                   [04]
                        0000    108         .LIBRARY        /$DS/                   ;                                   [04]
                        0000    109         .LIBRARY        /$DIAG/                 ;                                   [04]
                        0000    110 ;                                                                                   [04]
                        0000    111 ; Macros                                                                            [04]
                        0000    112 ;                                                                                   [04]
                        0000    113
                        0000    114 ;                                                                                   [04]
                        0000    115 ; Equated Symbols                                                                   [04]
                        0000    116 ;                                                                                   [04]
                        0000    117
                        C000    118         CLIDEF
                        0000    119         $PRDEF
                        0000    120         $BTDDEF                                 ; VMS boot device codes             [07]
                        0000    121         $MSCPDef                                ; Define disk protocol              [07]
                        0000    122
                        0000    123 ;
                        0000    124 ; Define Ptable data structures
                        0000    125 ;
                        0000    126
                        0000    127         $DS_HPODEF
                        0000    128         $DS_RP04_DEF
                        0000    129         $DS_RK61T_DEF
                        0000    130         $ds_rb730_def
                        0000    131         $ds_r80_def
                        0000    132         $ds_rl02_def
                        0000    133         $DS_RK06_DEF
                        0000    134         $DS_RL11_DEF
                        0000    135         $DS_RL01_DEF
                        0000    136         $Ds_UDA50_Def                           ;                                   [07]
                        0000    137         $Ds_RA80_Def                            ;                                   [07]
                        0000    138         $DS_CI_DEF                              ;                                   [10]
                        0000    139         $DS_DISK_DEF                            ;                                   [13]
                        0000    140
                        0000    141 ;
                        0000    142 ; Define miscellaneous structures/constants
                        0000    143 ;
                        0000    144
                        0000    145         $DS_DSADEF
                        0000    146         $DS_DSDEF
                        0000    147         $SSDef                                  ; Define SS status codes            [07]
                        0000    148
                        0000    149         $DS_CHCDEF
                        0000    150         $DS_CHIDEF
                        0000    151
00000C00                0000    152 RK_CS1=0                    ; Offset to control/status reg 1
00000040                0000    153 RK_CS1_M_IE=1@6             ; Interrupt bit
00000080                0000    154 RK_CS1_M_RDY=1@7            ; Controller ready
00008000                0000    155 RK_CS1_M_CERR=1@15          ; Clear controller errors
00000008                0000    156 RK_CS2=8                    ; Offset to control/status reg 2
0000000A                0000    157 RK_DS=10                    ; Offset to Drive status register
00000008                0000    158 RK_DS_V_DDT=8               ; Zero if RK06 drive, 1 if RK07
00000004                0000    159 F_DRVCLR=4                  ; Function to clear indicated drive
                        0000    160
```

```
                        0000    161
                00000000 0000   162 RL_CS=0                     ; offset to control/status register
                00000080 0000   163 RL_CS_M_CRDY=1@7            ; controller ready
                00000040 0000   164 RL_CS_M_IE=1@6              ; interrupt bit
                00000004 0000   165 RL_DA=4                     ; offset to disk address register
                00000001 0000   166 RL_DA_M_MRK=1@0             ; bit 0 in disk address register
                00000002 0000   167 RL_DA_M_STS=1@1             ; get status bit
                00000008 0000   168 RL_DA_M_RST=1@3             ; reset bit in disk adress register
                00000006 0000   169 RL_MP=6                     ; offset to multipurpose register
                00000007 0000   170 RL_MP_V_TYP=7               ; 0=RL01,1=RL02
                00000004 0000   171 F_STAT=4                    ; get status function
                         0000   172
                00000000 0000   173 RP_CS1=0                    ; Offset to control/status reg 1
                00000000 0000   174 F_NOP=0                     ; Function for drive NOP
                00000018 0000   175 RP_DT=24                    ; Offset to drive type register
                00000000 0000   176 RP_DT_V_DTN=0               ; Low bit of drive type
                00000009 0000   177 RP_DT_S_DTN=9               ; Size of drive type
                00000010 0000   178 DT_RP04=^X10                ; RP04 drive type
                00000011 0000   179 DT_RP05=^X11                ; RP05
                00000012 0000   180 DT_RP06=^X12                ; RP06
                00000021 0000   181 DT_RP07_HPT=^X21            ; RP07(with head per track option)
                00000022 0000   182 DT_RP07=^X22                ; RP07(w/o head per track option)
                00000015 0000   183 DT_RM03=^X15                ; RM03
                00000016 0000   184 DT_RM80=^X16                ; RM80
                00000017 0000   185 DT_RM05=^X17                ; RM05
                         0000   186
                         0000   187 ;
                         0000   188 ; RB730:/RB02/RB80 REGISTER OFFSETS FROM CSR ADDRESS
                         0000   189 ;
                         0000   190         $DEFINI RB                      ; START OF REGISTER DEFINITIONS
                         0000   191
                         0000   192 $DEF     RB_CS           .BLKL   1       ;CONTROL STATUS REGISTER (CSR)
                         0004   193         _VIELD  RB_CS,0,<-              ;START OF CSR BIT DEFINITIONS
                         0004   194                 <DRDY,,M>,-             ; DRIVE READY
                         0004   195                 <FCODE,3>,-             ; FUNCTION CODE
                         0004   196                 <,2>,-                  ; RESERVED BITS
                         0004   197                 <IE,,M>,-               ; INTERRUPT ENABLE
                         0004   198                 <CRDY,,M>,-             ; CONTROLLER READY
                         0004   199                 <DS,2>,-                ; DRIVE SELECT
                         0004   200                 <OPI,,M>,-              ; OPERATION INCOMPLETE
                         0004   201                 <CRC,,M>,-              ; DATA CRC OR HEADER CRC OR DATA ECC
                         0004   202                 <DLT,,M>,-              ; DATA LATE OR HEADER NOT FOUND
                         0004   203                 <NXM,,M>,-              ; NON-EXISTENT MEMORY
                         0004   204                 <DE,,M>,-               ; DRIVE ERROR
                         0004   205                 <CE,,M>,-               ; COMPOSITE ERROR
                         0004   206                 <ATN,4>,-               ; DRIVE ATTENTION BITS
                         0004   207                 <ECC,2>,-               ; ECC STATUS
                         0004   208                 <SSEI,,M>,-             ; SKIP SECTOR ERROR INHIBIT
                         0004   209                 <SSE,,M>,-              ; SKIP SECTOR ERROR
                         0004   210                 <IR,,M>,-               ; IDC INTERRUPT REQUEST
                         0004   211                 <MTN,,M>,-              ; MAINTENANCE MODE
                         0004   212                 <TYP,,M>,-              ; DRIVE TYPE 1=RB80, 0=RB02
                         0004   213                 <ASSI,,M>,-             ; AUTOMATIC SKIP SECTOR INHIBIT
                         0004   214                 <TOI,,M>,-              ; TIME OUT INHIBIT (U-DIAG'S)
                         0004   215                 <FMT,,M>,-              ; R80 FORMAT CONTROL
                         0004   216                 <,2>-                   ; RESERVED BITS
                         0004   217         >                               ;END CSR BIT DEFINITIONS
```

```
                        0004    218
                        0004    219
                        0004    220 $DEF    RB_BA           .BLKL   1       ;BUS ADDRESS REGISTER (BAR)
                        0008    221
                        0008    222 $DEF    RB_BC           .BLKL   1       ;BYTE COUNT REGISTER (BCR)
                        000C    223
                        000C    224 $DEF    RB_DA           .BLKL   1       ;DISK ADDRESS REGISTER (DAR)
                        0010    225         _VIELD  RB_DA,0,<-              ;START OF DAR BIT DEFINITIONS
                        0010    226                 <SEC,8>,-               ; SECTOR
                        0010    227                 <TRK,8>,-               ; TRACK
                        0010    228                 <CYL,16>-               ; CYLINDER
                        0010    229                 >                       ;END OF DAR BIT DEFINITIONS
                        0010    230
                        0010    231 $DEF    RB_MP           .BLKL   1       ;MULTIPURPOSE REGISTER (MPR)
                        C014    232         _VIELD  RB_MP,0,<-              ;RB02 STATUS WORD DEFINITIONS
                        0014    233                 <STA,3>,-               ; DRIVE STATE
                        0014    234                 <BH,,M>,-               ; BRUSH HOME
                        0014    235                 <HO,,M>,-               ; HEADS OUT
                        0014    236                 <CO,,M>,-               ; COVER OPEN
                        0014    237                 <HS,,M>,-               ; HEAD SELECT
                        0014    238                 <,1>,-                  ; RESERVED
                        0014    239                 <DSE,,M>,-              ; DRIVE SELECT ERROR
                        0014    240                 <VC,,M>,-               ; VOLUME CHECK
                        C014    241                 <WGE,,M>,-              ; WRITE GATE ERROR
                        0014    242                 <SPE,,M>,-              ; SPIN ERROR
                        0014    243                 <SKTO,,M>,-             ; SEEK TIME OUT
                        0014    244                 <WL,,M>,-               ; WRITE LOCK
                        0014    245                 <CHE,,M>,-              ; CURRENT HEAD ERROR
                        0014    246                 <WDE,,M>-               ; WRITE DATA ERROR
                        0014    247                 >                       ;
                        0014    248         _VIELD  RB_MP,0,<-              ;GET STATUS COMMAND DEFINITIONS
                        0014    249                 <MRK,,M>,-              ; MARK (ALWAYS 1)
                        0014    250                 <STS,,M>,-              ; GET STATUS
                        0014    251                 <,1>,-                  ; RESERVED
                        0014    252                 <RST,,M>,-              ; RESET
                        0014    253                 >
                        0014    254         _VIELD  RB_MP,0,<-              ;RB80 STATUS WORD DEFINITIONS
                        0014    255                 <SEC,5>,-               ; CURRENT RB80 SECTOR
                        0014    256                 <,3>,-                  ; RESERVED
                        0014    257                 <FLT,,M>,-              ; DRIVE FAULT
                        0014    258                 <PLGV,,M>,-             ; PLUG VALID
                        0014    259                 <SKE,,M>,-              ; SEEK ERROR
                        0014    260                 <ONCY,,M>,-             ; ON CYLINDER
                        0014    261                 <DRDY,,M>,-             ; DRIVE READY
                        0014    262                 <WTP,,M>,-              ; WRITE PROTECT
                        0014    263                 <,2>,-                  ; RESERVED
                        0014    264                 >                       ;END MPR BIT DEFINITIONS
                        0014    265
                        0014    266         $DEFEND RB                      ;END RB730:RB80/RB02 REGISTER DEFS
                        0000    267
                        0000    268 ;
                        0000    269 ; HARDWARE FUNCTION CODES
                        0000    270 ;
                        0000    271
                00000004 0000   272 F_GETSTATUS=2*2                         ;GET STATUS
                        0000    273
                        0000    274 ;                                                                   [04]
```

```
                                0000   275 ; External Symbols                                                                  [04]
                                0000   276 ;                                                                                  [04]
                                0000   277         .EXTRN  SCB_BASE, SCB_IMAGE, IOC$K_IOSPACE
                                0000   278         .EXTRN  EXE$ALONONPAGED, EXE$DEANONPAGED
                                0000   279         .EXTRN  DS$WAITUS, DS_ERRSUP, INS$PTABLE, DSP$CONFIG_ADP, DS$CHANNEL
                                0000   280         .EXTRN  DSV$SETLOAD
                                0000   281         .EXTRN  SYS$UNWIND
                                0000   282         .EXTRN  DS$GA_SELECTED, DS$GL_CLIBASE
                                0000   283         .Extrn  Uda_Response, UD_Init           ; UDA-50 data block/init code   [07]
                                0000   284         .Extrn  Dsr$Allocate_Pseudo_RPB         ;                               [07]
                                0000   285         .Extrn  Dsr$DeAllocate_Pseudo_RPB       ;                               [07]
                                0000   286         .EXTRN  BOOT$L_ADP,BOOT$L_R2,BOOT$L_DEVTYP,BOOT$L_UNIT,BOOT$L_CSR      ;[15]
                                0000   287
                                0000   288
                            0000C000   289         .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                               [04]
                                0000   290        ModNam  CONFIG
                                0007   291 T_TYPES:
            41 42 44 5F 34 30 50 52     0007   292         .ASCII  "RP04_DBA"      ; ^^10  RP04
            41 42 44 5F 35 30 50 52     000F   293         .ASCII  "RP05_DBA"      ; ^X11  RP05
            41 42 44 5F 36 30 50 52     0017   294         .ASCII  "RP06_DBA"      ; ^X12  RP06
                    00000000 00000000   001F   295         .QUAD   0               ; ^X13
            41 52 44 5F 33 30 4D 52     0027   296         .ASCII  "RM03_DRA"      ; ^X14  RM03 drive
                    00000000 00000000   002F   297         .QUAD   0
            41 52 44 5F 30 38 4D 52     0037   298         .ASCII  "RM80_DRA"      ; ^X16 RM80
            41 52 44 5F 35 30 4D 52     003F   299         .ASCII  "RM05_DRA"      ; ^X17 RM05
00000000'00000000'00000000'00000000'    0047   300         .LONG   0[18]           ; 9 Unused Quadwords
00000000'00000000'00000000'00000000'    0057
00000000'00000000'00000000'00000000'    0067
00000000'00000000'00000000'00000000'    0077
                    00000000'00000000'  0087
            41 52 44 5F 37 30 50 52     008F   301         .ASCII  "RP07_DRA"      ; ^X21 RP07(with head per track option)
            41 52 44 5F 37 30 50 52     0097   302         .ASCII  "RP07_DRA"      ; ^X22 RP07(w/o head  per track option)
                                        009F   303
                            00000013    009F   304 K_TYPES=<.-T_TYPES>/8           ; number of entries
```

```
                                009F    306          .SBTTL  INI$LOAD_DEVICE          Make ptables for console and boot devices
                             00000000   307          .PSECT  Code, Exe, Shr, NoWrt, Long
                                0000    308 ;++                                                                          begin   [15]
                                0000    309 ; FUNCTIONAL DESCRIPTION:
                                0000    310 ;
                                0000    311 ;       This routine sets up the necessary P-tables
                                0000    312 ;       for the console load device and the boot device.
                                0000    313 ;
                                0000    314 ; CALLING SEQUENCE:
                                0000    315 ;
                                0000    316 ;       JSB     INI$LOAD_DEVICE
                                0000    317 ;
                                0000    318 ; INPUT PARAMETERS:
                                0000    319 ;
                                0000    320 ;       None
                                0000    321 ;
                                0000    322 ; IMPLICIT INPUTS:      NONE
                                0000    323 ;
                                0000    324 ; OUTPUT PARAMETERS:    NONE
                                0000    325 ;
                                0000    326 ; IMPLICIT OUTPUTS:
                                0000    327 ;
                                0000    328 ;       P-Tables for each device in the boot path and the
                                0000    329 ;       console load device.
                                0000    330 ;
                                0000    331 ; COMPLETION CODES:
                                0000    332 ;
                                0000    333 ;       None
                                0000    334 ;
                                0000    335 ; SIDE EFFECTS:
                                0000    336 ;
                                0000    337 ;       None
                                0000    338 ;
                                0000    339 ;--
                                0000    340
                                0000    341 INI$LOAD_DEVICE::
                                0000    342                                                  ; Build the console ptable
                    51    32    3C    0000    343          MovZWL  #Hp$A_Dependent, R1     ; Length of Ptable to make
               00000000'EF    16    0003    344          Jsb     L^Exe$AloNonPaged       ; Allocate it
                     48    50    E9    0009    345          BlbC    R0, 10$                 ; Skip setup if couldn't do it
                           3E    BB    000C    346          PushR   #^M<R1,R2,R3,R4,R5>     ; Save volatile registers
 62    51    00    6E    00    2C    000E    347          MovC5   #0, (Sp), #0, R1, (R2)  ; Clear the Ptable
                           3E    BA    0014    348          PopR    #^M<R1,R2,R3,R4,R5>     ; Restore registers
                  08    A2    51    B0    0016    349          MovW    R1, Hp$W_Size(R2)       ; Set size field
                        62    05    D0    001A    350          MovL    #5, Hp$Q_Device(R2)     ; Set length of descriptor
                  50      0C  A2    9E    001D    351          MovAB   Hp$T_Device(R2), R0     ; Set address of ''_CSA0''
                  04    A2      60    9E    0021    352          MovAB   (R0), Hp$Q_Device+4(R2) ; Set address of string
              80  4153435F    8F    D0    0025    353          MovL    #^A''_CSA'', (R0)+      ; Set first 4 chars of name
                        60      30    90    002C    354          MovB    #^A''0'', (R0)          ; Set last character
                  50      26  A2    9E    002F    355          MovAB   Hp$T_Type(R2), R0       ; Set address of type string
              80  6E6F6307    8F    D0    0033    356          MovL    #7!<2^A''con''>@8>, (R0)+ ; Set length/''con''
              60  656C6F73    8F    D0    003A    357          MovL    #^A''sole'', (R0)       ; Set rest of name
              00000000'EF       62    FA    0041    358          CallG   (R2), L^Ins$Ptable      ; Insert the Ptable in list
                           09   50    E8    0048    359          BlbS    R0, 10$                 ; Continue if it worked
                        50      52    D0    004B    360          MovL    R2, R0                  ; Else, copy length to R0
               00000000'EF      16    004E    361          Jsb     L^Exe$DeaNonPaged       ; .. and deallocate the buffer
                                0054    362 10$:
```

```
                50    00000000'EF   D0  0054    363          MOVL    L^BOOT$L_ADP,R0                 ; Get boot adapter address
                                21   13  005B    364          BEQL    20$                             ; Branch if not there
                      00000000'EF   DD  005D    365          PUSHL   L^BOOT$L_R2                     ; CI station number
                      00000000'EF   DD  0063    366          PUSHL   L^BOOT$L_DEVTYP                 ; Push boot device type
                      00000000'EF   DD  0069    367          PUSHL   L^BOOT$L_UNIT                   ; Push BOOT device unit number
                      00000000'EF   DD  006F    368          PUSHL   L^BOOT$L_CSR                    ; Push BOOT device CSR address
                                50  DD  0075    369          PUSHL   R0                              ; Push BOOT adapter address
                0000007F'EF   05  FB  0077    370          CALLS   #5, L^DSP$CONFIG_LOAD           ; Auto-config load device
                                      007E    371 20$:
                                05    007E    372          RSB                                     ;                               end     [15]
```

```
                     007F    374          .List   MEB
                     007F    375          .SBTTL  DSP$CONFIG_LOAD Configure load device
                     007F    376 ;++
                     007F    377 ; FUNCTIONAL DESCRIPTION:
                     007F    378 ;
                     007F    379 ;       This routine determines sets up the necessary P-tables
                     007F    380 ;       for the load device given the adapter configuration register,
                     007F    381 ;       the device register and unit number.
                     007F    382 ;
                     007F    383 ; CALLING SEQUENCE:
                     007F    384 ;
                     007F    385 ;       DSP$CONFIG_LOAD(Adp adr, Dev adr, Unit)
                     007F    386 ;
                     007F    387 ; INPUT PARAMETERS:
                     C07F    388 ;
                     007F    389 ;       4(AP)   Address of configuration status register of adapter
                     007F    390 ;       8(AP)   Address of CSR for device.
                     007F    391 ;       12(AP)  Unit number of drive
                     007F    392 ;       16(AP)  device type: 0=massbus,1=RK06/07,2=RL01/02,20=HSC50
                     007F    393 ;       20(ap)  CI-HSC50 port station number
                     007F    394 ;
                     007F    395 ; IMPLICIT INPUTS:      NONE
                     007F    396 ;
                     007F    397 ; OUTPUT PARAMETERS:    NONE
                     007F    398 ;
                     007F    399 ; IMPLICIT OUTPUTS:
                     007F    400 ;
                     007F    401 ;       P-Tables for each device in the load control and data path
                     007F    402 ;
                     007F    403 ; COMPLETION CODES:
                     007F    404 ;
                     007F    405 ;       SS$_NORMAL
                     007F    406 ;
                     007F    407 ; SIDE EFFECTS:
                     007F    408 ;
                     007F    409 ;       The device and the adapter specified are pinged to determine
                     007F    410 ;       interrupt level and interrupt vectors.
                     007F    411 ;
                     007F    412 ; REGISTER USAGE:
                     007F    413 ;
                     007F    414 ;       R4      Address of P-table being created
                     007F    415 ;       R5      Address of channel or device
                     007F    416 ;--
                     007F    417          .NLIST  ME,MEB
                     007F    418          $OFFSET 0,NEGATIVE, < -
                     007F    419                  <L_STATUS,8>,-
                     007F    420                  <L_LINK,4>,-
                     007F    421                  <L_TYPE,4>,-
                     007F    422                  <L_LOCAL,0>>
                     FFF8        L_STATUS:
                     FFF4        L_LINK:
                     FFF0        L_TYPE:
                     FFF0        L_LOCAL:
                     007F    423          .LIST   MEB
```

```
                          023C  007F    425 .ENTRY  DSP$CONFIG_LOAD,^M<R2,R3,R4,R5,R9>      ;                                [07]
                                0081    426
          5E    FO AD     9E  0081    427         MOVAB   L_LOCAL(FP),SP              ; Allocate space for local storage
                FO AD     DF  0085    428         PUSHAL  L_TYPE(FP)                 ; Address to store type
                F4 AD     DF  0088    429         PUSHAL  L_LINK(FP)                 ; Address of P-table created
                04 AC     DD  008B    430         PUSHL   4(AP)                      ; Address of Adapter config reg
  00000000'EF     03     FB  008E    431         CALLS   #3,L^DSP$CONFIG_ADP        ; Configure the adaptor          [05]
                37 50     E9  0095    432         BLBC    R0,10$                     ; Return on error
                                0098    433
          11    10 AC     D1  0098    434         CmpL    16(Ap), #Btd$K_UDA         ; Is it a UDA (code 17)          [07]
                   03     12  009C    435         BNeq    5$                         ; If not, do CASE for contiguous codes [07]
                 01D1     31  009E    436         BrW     Con_Uba                    ; UDA50 sits on UNIBUS           [07]
          20    10 AC     D1  00A1    437 5$:     CMPL    16(AP),#BTD$K_HSCCI        ; IS IT A HSC50 ?                [10]
                   03     12  00A5    438         BNEQ    7$                         ; BRANCH IF NOT.                 [10]
                 0026     31  00A7    439         BRW     CON_CI_HSCCI               ; BRANCH FOR HSC50               [10]
                          00AA    440 7$:     CASE    16(AP),TYPE=L,LIMIT=#0, -          ;                            [07]
                          00AA    441                 displist=<con_mba,con_uba,con_uba,con_uba>
    03'   00    10 AC     CF  00AA            CASEL   16(AP),#0,S^#<<30001$-30000$>/2>-1
                          00AF        30000$:
                0133'     00AF                .SIGNED_WORD    con_mba-30000$
                01C3'     00B1                .SIGNED_WORD    con_uba-30000$
                01C3'     00B3                .SIGNED_WORD    con_uba-30000$
                01C3'     00B5                .SIGNED_WORD    con_uba-30000$
                          00B7        30001$:
                          00B7    442         ERRSUP_S                           ; Should never happen!
      00000000'EF    DF   00B7                        PUSHAL  $MODULE
                     00   00BD                        PUSHL   #0
                     01   00BF                        PUSHL   #$ER
  00000000'EF        03   00C1                        CALLS   #3, DS_ERRSUP
50    00660002 8F    D0   00C8    443         MOVL    #DS$_ERROR,R0              ; Flag error
                          00CF    444 10$:
                     04   00CF    445         RET
```

```
                                00D0   447 CON_CI_HSCCI:
                                00D0   448 ;+
                                00D0   449 ; BUILD PT FOR HSC50 DEVICE
                                00D0   450 ;-
          51   00000041 8F    D0  00D0   451          MOVL    #HP$K_CI_LEN, R1 ; LENGTH OF PT FOR CI NODE.                    [10]
                     031B 30    00D7   452          BSBW    ALLOCATE_PTABLE ; ALLOCATE THE SPACE FOR THIS DRIVE.            [10]
                   01 50  E8    00DA   453          BLBS    R0,10$          ; BRANCH IF SUCCESS                             [10]
                          04    00DD   454          RET                     ; RETURN ON ERROR                               [10]
                    08 A2   51  B0  00DE   455 10$:   MOVW    R1,HP$W_SIZE(R2) ; SET SIZE OF STRUCTURE.                       [10]
                    26 A2   07  90  00E2   456          MOVB    #7,HP$T_TYPE(R2) ; SET LENGTH OF TYPE NAME                      [10]
 27 A2   2045444F 4E5F4943 8F  7D  00E6   457          MOVQ    #^A"CI_NODE ",HP$T_TYPE+1(R2) ; SET DEVICE TYPE                 [10]
                       62   05  D0  00F2   458          MOVL    #5,HP$Q_DEVICE(R2) ; SET LENGTH OF NAME                         [10]
                    50   0C A2  9E  00F5   459          MOVAB   HP$T_DEVICE(R2),R0 ; ADDRESS OF DEVICE NAME                     [10]
                    04 A2   60  9E  00F9   460          MOVAB   (R0),HP$Q_DEVICE+4(R2) ; SET ADDRESS OF DEVICE NAME             [10]
          80   4149435F 8F  D0  00FD   461          MOVL    #^A"_CIA",(R0)+ ; INSERT "_CIA".                                [10]
                    53   14 AC  D0  0104   462          MOVL    20(AP),R3       ; GET PORT NUMBER                               [11]
                         54   D4  0108   463          CLRL    R4              ; CLEAR HIGH ORDER FIELD                        [11]
    54   53   53   00000064 8F  7B  010A   464          EDIV    #100,R3,R3,R4   ; SET FIRST DIGIT                               [10]
                         06   13  0113   465          BEQL    20$             ; BRANCH IF NOT THIS LARGE                      [10]
                80   53   30  81  0115   466          ADDB3   #^A"0",R3,(R0)+ ; SET FIRST ASCII DIGIT                         [10]
                         62   D6  0119   467          INCL    HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                     [10]
                         55   D4  011B   468 20$:      CLRL    R5              ; CLEAR HIGH ORDER FIELD                        [11]
             54   53   54   0A  7B  011D   469          EDIV    #10,R4,R3,R4    ; GET NEXT DIGIT                                [10]
                         06   13  0122   470          BEQL    30$             ; BRANCH IF NOT THIS LARGE                      [10]
                80   53   30  81  0124   471          ADDB3   #^A"0",R3,(R0)+ ; SET THIS ASCII DIGIT                          [10]
                         62   D6  0128   472          INCL    HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                     [10]
                60   54   30  81  012A   473 30$:      ADDB3   #^A"0",R4,(R0)  ; SET LAST DIGIT                                [10]
              0B A2   14 AC  90  012E   474          MOVB    20(AP),HP$B_DRIVE(R2)   ; SET port NUMBER                       [13]
 18 A2   04 AC   00000000'8F  C9  0133   475          BISL3   #<IOC$K_IOSPACE>,4(AP),HP$A_DEVICE(R2) ; SET DEVICE ADDR         [10]
              1C A2   18 A2  D0  013D   476          MOVL    HP$A_DEVICE(R2),HP$A_DVA(R2) ; Set direct address               [13]
              20 A2   F4 AD  D0  0142   477          MOVL    L_LINK(FP),HP$A_LINK(R2) ; SET LINK ADDRESS.                    [10]
                3D A2   04  9A  0147   478          MOVZBL  #^X4,HP$L_CI_INDEX(R2)  ; SET INDEX USED BY $DS_CASE            [10]
          39 A2   80000004 8F  D0  0148   479          MOVL    #^x80000004,HP$L_CI_MAINT_ID(R2) ; SET HSC50 IDENTIFICATION     [10]
          35 A2   4F710200 8F  D0  0153   480          MOVL    #^X4F710200,HP$L_CI_FUNC(R2) ; FUNCTIONALITY OF HSC50           [10]
              34 A2   14 AC  90  015B   481          MOVB    20(AP), HP$B_CI_NODE(R2)    ; SET NODE NUMBER.                  [10]
                         52   DD  0160   482          PUSHL   R2                      ; Save address of CI_NODE p-table       [13]
                0000'CF   62  FA  0162   483          CALLG   (R2), W^INS$PTABLE      ; Insert it in list                     [13]
                      03 50  E8  0167   484          BLBS    R0,100$                 ; Continue if no errors.                [13]
                       00F7  31  016A   485          BRW     CON_RELEASE_X           ; Go release buffer                     [13]
                    51   32  D0  016D   486 100$:    MOVL    #HP$K_DISK_LEN,R1       ; Length of ptable for DISK             [13]
                       0282  30  0170   487          BSBW    ALLOCATE_PTABLE         ; Allocate the space for this drive     [13]
                      01 50  E8  0173   488          BLBS    R0,110$                 ; Branch if success                     [13]
                             04  0176   489          RET                             ; Return on error.                      [13]
                    08 A2   51  B0  0177   490 110$:    MOVW    R1,HP$W_SIZE(R2)        ; Set size of structure                 [13]
                    26 A2   04  90  017B   491          MOVB    #4,HP$T_TYPE(R2)        ; Set length of type name               [13]
          27 A2   4B534944 8F  D0  017F   492          MOVL    #^A"DISK",HP$T_TYPE+1(R2) ; Set device type                     [13]
                       62   0B  D0  0187   493          MOVL    #11,HP$Q_DEVICE(R2)     ; Set  length of name                   [13]
                    50   0C A2  9E  018A   494          MOVAB   HP$T_DEVICE(R2),R0      ; Address of device name                [13]
                    04 A2   60  9E  018E   495          MOVAB   (R0),HP$Q_DEVICE+4(R2)  ; Set address of device name            [13]
             80   4353485F 8F  D0  0192   496          MOVL    #^A"_HSC",(R0)+         ; Insert "_HSC".                        [14]
                80   3035 8F  B0  0199   497          MOVW    #^A"50",(R0)+           ; Insert "50".                          [14]
             80   41554424 8F  D0  019E   498          MOVL    #^A"$DUA",(R0)+         ; Insert "$DUA".                        [13]
                    53   0C AC  D0  01A5   499          MOVL    12(AP),R3               ; Get drive #                           [17]
                         54   D4  01A9   500          CLRL    R4                      ; Clear high order field                [13]
    54   53   53   00000064 8F  7B  01AB   501          EDIV    #100,R3,R3,R4           ; Set first digit                       [13]
                         06   13  01B4   502          BEQL    120$                    ; Branch if number is <100              [13]
                80   53   30  81  01B6   503          ADDB3   #^A"0",R3,(R0)+         ; Set first ascii digit                 [13]
```

```
                        62    D6   01BA   504           INCL    HP$Q_DEVICE(R2)          ; Increase length of string          [13]
                  0A    0C AC   91   01BC   505 120$:    CMPB    12(AP),#10               ; Is it >=10    ?                    [17]
                        0D      19   01C0   506           BLSS    130$                     ; Branch if not.                     [17]
                        55      D4   01C2   507           CLRL    R5                       ; Clear high order field             [13]
              54  53    54      0A   7B   01C4   508           EDIV    #10,R4,R3,R4             ; Get next digit                     [13]
                  80    53      30   81   01C9   509           ADDB3   #^A'0',R3,(R0)+          ; Set this ascii digit               [13]
                        62      D6   01CD   510           INCL    HP$Q_DEVICE(R2)          ; Increase length of string          [13]
                  60    54      30   81   01CF   511 130$:    ADDB3   #^A'0',R4,(R0)           ; Set last digit                     [13]
                0B A2     0C AC   90   01D3   512           MOVB    12(AP),HP$B_DRIVE(R2)    ; Set drive number                   [13]
                          1C A2   D4   01D8   513           CLRL    HP$A_DVA(R2)             ; Clear direct address               [13]
                     20 A2    8E   D0   01DB   514           MOVL    (SP)+,HP$A_LINK(R2)      ; Set link address.                  [13]
                           01E2   31   01DF   515           BRW     CON_NORMAL_X             ; SET DEFAULT LOAD DEVICE.           [10]
                                       01E2   516
```

```
                                01E2   518 CON_MBA:
                                01E2   519 ;+
                                01E2   520 ; BUILD PT FOR MASSBUS DRIVE
                                01E2   521 ;-
     55   04 AC   00000400'8F   C9  01E2   522          BISL3   #<IOC$K_IOSPACE>+<^X400>,4(AP),R5      ; Base adapter
          55   03   07   0C AC   F0  01EB   523          INSV    12(AP),#7,#3,R5                      ; Insert unit number to make drive csr
                         65   01   D0  01F1   524          MOVL    #F_NOP!1,RP_CS1(R5)                  ; Issue NOP to drive
                       50   18 A5   D0  01F4   525          MOVL    RP_DT(R5),R0                         ; Get drive type register
                    50   09   00   EF  01F8   526          EXTZV   #RP_DT_V_DTN,#RP_DT_S_DTN, -
                                7E       01FC   527                  R0,-(SP)                         ; Isolate drive type
                         51   32   D0  01FD   528          MOVL    #RP04$K_LEN,R1                       ; Length of PT for RP04,RP05,RP06,RM03
                              01F2   30  0200   529          BSBW    Allocate_Ptable                      ; Allocate the space for this device    [07]
                                53 8ED0  0203   530          POPL    R3                                   ; Get drive type back
                              01 50   E8  0206   531          BLBS    R0,10$                               ; Branch if success
                                      04  0209   532          RET                                  ; Return on error
                                      020A   533 10$:
                       08 A2   51   B0  020A   534          MOVW    R1,HP$W_SIZE(R2)                     ; Set size of structure
                                      020E   535
                         53   10   C2  020E   536          SUBL    #DT_RP04,R3                          ; Make zero correspond to RP04 drivetype
                                0F   19  0211   537          BLSS    30$                                  ; Branch if less than that
                         53   13   D1  0213   538          CMPL    #K_TYPES,R3                          ; Range check drive type
                                0A   15  0216   539          BLEQ    30$                                  ; Branch if unfamilar drive type
               50   00000007'EF43   7D  0218   540          MOVQ    L^T_TYPES[R3],R0                     ; Get device type and name              [05]
                                13   12  0220   541          BNEQ    40$                                  ; Branch if valid
                                      0222   542 30$:     ERRSUP_S                                     ; Unknown drive type
                    00000000'EF   DF  0222                              PUSHAL  $MODULE
                                00   DD  0228                              PUSHL   #0
                                02   DD  022A                              PUSHL   #$ER
                00000000'EF     03   FB  022C                              CALLS   #3, DS_ERRSUP
                                2F   11  0233   543          BRB     CON_RELEASE_X                        ; Release buffer and exit
                                      0235   544
                    26 A2   04   90  0235   545 40$:     MOVB    #4,HP$T_TYPE(R2)                     ; Set length of type name
                    27 A2   50   D0  0239   546          MOVL    R0,HP$T_TYPE+1(R2)                   ; Set drive type
                                      023D   547
                         62   05   D0  023D   548          MOVL    #5,HP$Q_DEVICE(R2)                   ; Set length of name
                    50   0C A2   9E  0240   549          MOVAB   HP$T_DEVICE(R2),R0                   ; Address of device name
                    04 A2   60   9E  0244   550          MOVAB   (R0),HP$Q_DEVICE+4(R2)               ; Set address of device name
                         80   51   D0  0248   551          MOVL    R1,(R0)+                             ; Insert "_xxx" xxx is DBA or DRA
              80   0C AC   30   81  024B   552          ADDB3   #^A'0',12(AP),(R0)+                  ; Set unit number
                08 A2   0C AC   90  0250   553          MOVB    12(AP),HP$B_DRIVE(R2)                ; Set drive number
                                      0255   554
                    18 A2   55   D0  0255   555          MOVL    R5,HP$A_DEVICE(R2)                   ; Set device address
                         1C A2   D4  0259   556          CLRL    HP$A_DVA(R2)                         ; Clear direct address
                20 A2   F4 AD   D0  025C   557          MOVL    L_LINK(FP),HP$A_LINK(R2)             ; Set link address
                          0160   31  0261   558          BRW     CON_NORMAL_X                         ; Set Default load device
                                      0264   559
                                      0264   560 CON_RELEASE_X:
                         50   52   D0  0264   561          MOVL    R2,R0                                ; Copy buffer address
                              FD96'  30  0267   562          BSBW    EXE$DEANONPAGED                      ; Release the buffer
                                      026A   563 CON_ERROR_X:
           50   00660002 8F   D0  026A   564          MOVL    #DS$_ERROR,R0                        ; Error
                                      0271   565 CON_X:
                                04  0271   566          RET
```

```
                                0272   568 CON_UBA:
                                0272   569 ;+
                                0272   570 ; initialize adapter and unibus
                                0272   571 ;-
      0000FE0C'EF    01    D0   0272   572          MOVL    #1,DSA$GL_UNITS          ; Set 1 UUT
  00000004'EF    F4 AD    D0   0279   573          MOVL    L_LINK(FP),DS$GA_SELECTED+4 ; Set address of P-table for channel
                                0281   574
   53 00000000'EF    9E   0281   575          MOVAB   DS$CHANNEL,R3            ; Point to channel services
         F8 AD    7F   0288   576          PUSHAQ  L_STATUS(FP)             ; Address of status quadword
      00000406'EF    9F   028B   577          PUSHAB  L^COMMON_INTERRUPT       ; interrupt address for RK0X & RL0X      [05]
       03    10 AC    D1   0291   578          CmpL    16(Ap), #3               ; Is the device an RB730?                [06]
             07    12   0295   579          BNeq    1$                       ; Branch if not                          [06]
   6E 00000518'EF    9E   0297   580          MovAB   L^RB730_INTERRUPT, (Sp)  ; Replace interrupt address              [06]
             00    DD   029E   581 1$:       PUSHL   #CHC$_INITA              ; Use INITA                              [06]
          7E    04    7D   02A0   582          MOVQ    #4,-(SP)                 ; Unit zero and arg count=4
          63    6E    FA   02A3   583          CALLG   (SP),(R3)                ; Init adapter
            C8 50    E9   02A6   584          BLBC    R0,CON_X                 ; Exit if error
       08 AE    01    D0   02A9   585          MOVL    #CHC$_INITB,8(SP)        ; Function INITB
          63    6E    FA   02AD   586          CALLG   (SP),(R3)                ; Init Unibus
            BE 50    E9   02B0   587 2$:       BLBC    R0,CON_X                 ; Exit if error
       08 AE    06    D0   02B3   588          MOVL    #CHC$_CLEAR,8(SP)        ; Function Clear adapter
          63    6E    FA   02B7   589          CALLG   (SP),(R3)                ; Clear status
            B4 50    E9   02BA   590          BLBC    R0,CON_X                 ; Exit if error
       08 AE    04    D0   02BD   591          MOVL    #CHC$_ENINT,8(SP)        ; Function enable interrupts
          63    6E    FA   02C1   592          CALLG   (SP),(R3)                ; Init unibus
            AA 50    E9   02C4   593          BLBC    R0,CON_X                 ; Return on error
55 08 AC 00000000'8F    C9   02C7   594          BISL3   #IOC$K_IOSPACE,8(AP),R5 ; Get real address (VA)           [07]
                                02D0   595
          11    10 AC    D1   02D0   596          CmpL    16(Ap), #Btd$K_UDA       ; Is it a UDA (code 17)          [07]
             03    12   02D4   597          BNeq    3$                       ; If not, do CASE for contiguous codes   [07]
           034D    31   02D6   598          BrW     Con_Uba_Uda              ; UDA50                                  [07]
                                02D9   599 3$:       CASE    16(AP),TYPE=L,LIMIT=#1,-
                                02D9   600                   displist=<con_uba_rk,con_uba_rl,con_uba_rb>
   02' 01    10 AC    CF   02D9                    CASEL   16(AP),#1,S^#2<30003$-30002$>/2>-1
                                02DE           30002$:
                          0006' 02DE                    .SIGNED_WORD    con_uba_rk-30002$
                          013A' 02E0                    .SIGNED_WORD    con_uba_rl-30002$
                          0253' 02E2                    .SIGNED_WORD    con_uba_rb-30002$
                                02E4           30003$:
                                02E4   601
```

```
                                02E4    603 .enabl  lsb
                                02E4    604 CON_UBA_RK:
                                02E4    605 ;+
                                02E4    606 ; now build PT for RK611
                                02E4    607 ;-
          65    8000 8F    B0   02E4    608         MOVW    #RK_CS1_M_CERR,RK_CS1(R5); Clear controller errors
        08 A5     0C AC    B0   02E9    609         MOVW    12(AP),RK_CS2(R5)       ; Set unit number of drive in question
          65    0045 8F    B0   02EE    610         MOVW    #RK_CS1_M_IE!F_DRVCLR!1, -
                                02F3    611                 RK_CS1(R5)              ; Enable interrupts and clear drive
                                02F3    612         $DS_WAITUS_S    #1000           ; Wait for it to finish
                      00    DD  02F3                PUSHL   #0
                000003E8 8F    DD  02F5                PUSHL   #1000
            00000000'9F    02    FB  02FB                CALLS   #2, @#DS$WAITUS
                54    0A A5    3C  0302    613         MOVZWL  RK_DS(R5),R4            ; Get drive status
                65    0040 8F    AA  0306    614         BICW    #RK_CS1_M_IE,RK_CS1(R5) ; Clear interrupt enable
                  08 AE    05    D0  030B    615         MOVL    #CHC$_DSINT,8(SP)       ; Change function to disable
                     63    6E    FA  030F    616         CALLG   (SP),(R3)               ; Disable interrupts
                        12 50    E9  0312    617         BLBC    R0,10$                  ; Return if error
                           54    B5  0315    618         TSTW    R4                      ; Status valid?
                           0F    18  0317    619         BGEQ    20$                     ; Error if not
      54    54    01    08    EF  0319    620         EXTZV   #RK_DS_V_DDT,#1,R4,R4   ; Isolate drive type, 0=RK06, 1=Rk07
                                031E    621
                        51    37  3C  031E    622         MOVZWL  #RK611$K_LEN,R1         ; Length of RK611 PT
                           00D1    30  0321    623         BSBW    Allocate_Ptable         ; Allocate space for same                [07]
                              04 50    E8  0324    624         BLBS    R0,25$                  ; Branch if succeeded
                                0327    625 10$:
                                   04   0327    626         RET                             ; Return error
                                FF3F    31  0328    627 20$:    BRW     CON_ERROR_X             ; Leap to error exit
                                0328    628 25$:
                     08 A2    51    B0  032B    629         MOVW    R1,HP$W_SIZE(R2)        ; Set size of structure
                                032F    630
                        62    04    D0  032F    631         MOVL    #4,HP$Q_DEVICE(R2)      ; Set length of "_DMA"
                     50    0C A2    9E  0332    632         MOVAB   HP$T_DEVICE(R2),R0      ; Set address of "_DMA"
                     04 A2    50    D0  0336    633         MOVL    R0,HP$Q_DEVICE+4(R2)    ; Set address of device name
            60    414D445F 8F    D0  033A    634         MOVL    #^A"_DMA",(R0)          ; Set "_DMA"
                     18 A2    55    D0  0341    635         MOVL    R5,HP$A_DEVICE(R2)      ; Set device address
                        1C A2    D4  0345    636         CLRL    HP$A_DVA(R2)            ; Clear virtural address
                  20 A2    F4 AD    D0  0348    637         MOVL    L_LINK(FP),HP$A_LINK(R2); Set link to channel
               26 A2    5205 8F    B0  034D    638         MOVW    #^X5205,HP$T_TYPE(R2)   ; Set length and "R" of RK611
            28 A2    3131364B 8F    D0  0353    639         MOVL    #^A'K611',HP$T_TYPE+2(R2) ; Set "K611"
         32 A2    18 A2    12    00  EF  0358    640         EXTZV   #0,#18, HP$A_DEVICE(R2), -
                                0362    641                 RK611$L_CSR(R2)         ; Set CSR address
                                0362    642
                  24 A2    FE AD    B0  0362    643         MOVW    L_STATUS+6(FP), -
                                0367    644                 HP$W_VECTOR(R2)         ; Set current vector
                                   14   0367    645         BEQL    30$                     ; Branch if invalid vector
            50    FC AD    04    02  EF  0369    646         EXTZV   #CHI$V_IPL,#4, -
                                036F    647                 L_STATUS+4(FP),R0       ; Extract BR level of device
                     36 A2    50    90  036F    648         MOVB    R0,RK611$B_BR(R2)       ; Save BR level
                                0373    649
                00000000'EF    62    FA  0373    650         CALLG   (R2),L^INS$PTABLE       ; Add this PT                            [05]
                              03 50    E8  037A    651         BLBS    R0,40$                  ; Branch if success
                                FEE4    31  037D    652 30$:    BRW     CON_RELEASE_X           ; Release buffer and exit
                                0380    653 40$:
                  F4 AD    52    D0  0380    654         MOVL    R2,L_LINK(FP)           ; Save PT of RK611
                        51    32  3C  0384    655         MOVZWL  #RK06$K_LEN,R1          ; Length of RK06/RK07 PT
                           6C    10  0387    656         BSBB    Allocate_Ptable         ; Allocate space for PT                  [07]
```

```
                    9B 50    E9  0389    657          BLBC    R0,10$                          ; Take error exit
               08 A2   51    B0  038C    658          MOVW    R1,HP$W_SIZE(R2)                ; Set size of structure
                                 0390    659
                  62   05    D0  0390    660          MOVL    #5,HP$Q_DEVICE(R2)              ; Insert length of "_DMA0"
               50   0C A2    9E  0393    661          MOVAB   HP$T_DEVICE(R2),R0              ; Insert address of "_DMA0"
               04 A2   50    D0  0397    662          MOVL    R0,HP$Q_DEVICE+4(R2)            ; Set address of device name
            80   414D445F    8F  D0      039B    663          MOVL    #^A"_DMA",(R0)+                 ; Insert "_DMA"
             80   0C AC     30  81      03A2    664          ADDB3   #^A'0',12(AP),(R0)+             ; Insert unit number
               0B A2   0C AC     90      03A7    665          MOVB    12(AP),HP$B_DRIVE(R2)           ; Set drive number
                       18 A2     D4      03AC    666          CLRL    HP$A_DEVICE(R2)                 ; Clear device address
                       1C A2     D4      03AF    667          CLRL    HP$A_DVA(R2)                    ; Clear virtural address
                  20 A2   F4 AD     D0      03B2    668          MOVL    L_LINK(FP),HP$A_LINK(R2)        ; Link to RK611
            26 A2   304B5204    8F  D0      03B7    669          MOVL    #<^A"@RK0"-60>,-
                                                    03BF    670                                          HP$T_TYPE(R2)           ; Set length=4 and "RK0"
             2A A2   54   36     81      03BF    671          ADDB3   #^A'6',R4,HP$T_TYPE+4(R2); Set "6" or "7" if R4=1
                                                    03C4    672 CON_NORMAL_X:
                0000'CF   62     FA      03C4    673          callg   (r2), w^ins$ptable      ; Insert it in list
                       B1 50     E9      03C9    674          blbc    r0, 30$                 ; fail if error
                     50   52     D0      03CC    675          MOVL    R2,R0                   ; Copy address of P-table of load dev
            52   00000000'EF     DE      03CF    676          MOVAL   DS$GL_CLIBASE,R2        ; Set R2 to base of CLI data area
               08 A2   60     D0      03D6    677          MOVL    HP$Q_DEVICE(R0), -
                                                    03DA    678                                          CLI$Q_FILE(R2)          ; Set length of load device name
                  7E   3A     90      03DA    679          MOVB    #^A":",-(SP)            ; Tack a ":" to the end
                  53   60     D0      03DD    680          MOVL    HP$Q_DEVICE(R0),R3      ; Length of name. Any size.             [12]
                  53     D7      03E0    681          DECL    R3                      ; "_" not included                      [12]
               7E   0C A043     90      03E2    682 50$:     MOVB    HP$T_DEVICE(R0)[R3],-(SP)       ; Place name on stack   [12]
                     F8 53     F5      03E7    683          SOBGTR  R3,50$                  ; Loop to move all of name              [12]
                  0C A2   6E     9E      03EA    684          MOVAB   (SP),CLI$Q_FILE+4(R2)   ; Set address of load device name
                        FC0F'     30      03EE    685          BSBW    DSV$SETLOAD             ; Set default load device
                     50   01     D0      03F1    686          MOVL    #SS$_NORMAL,R0          ; Success
                                 04      03F4    687          RET
                                         03F5    688 .dsabl   lsb
```

```
                                03F5    690 .Subtitle       Common routines
                                03F5    691
                                03F5    692 ;
                                03F5    693 ; This little routine allocates a Ptable and pre-clears it.
                                03F5    694 ;
                                03F5    695 ; Inputs:
                                03F5    696 ;  R1 => Size of Ptable to allocate
                                03F5    697 ;
                                03F5    698 ; Outputs:
                                03F5    699 ;  R0 => Status code
                                03F5    700 ;  R1 => Size of Ptable
                                03F5    701 ;  R2 => Address of Ptable
                                03F5    702 ;
                                03F5    703
                                03F5    704 Allocate_Ptable:                            ; Use to allocate/clear Ptable          [07]
                00000000'EF   16 03F5    705         Jsb     L^Exe$AloNonPaged           ; Allocate the structure                [07]
                          3F  BB 03F8    706         PushR   #^M<R0,R1,R2,R3,R4,R5>      ; Save volatile registers               [07]
          62    51    00  6E  2C 03FD    707         MovC5   #0, (Sp), #0, R1, (R2)      ; Clear the Ptable                      [07]
                          3F  BA 0403    708         PopR    #^M<R0,R1,R2,R3,R4,R5>      ; Restore the registers                 [07]
                          05  0405    709         Rsb                                 ;                                       [07]
                              0406    710
                              0406    711 COMMON_INTERRUPT:
                          03  BB 0406    712         PUSHR   #^M<R0,R1>                  ; Save volatile regs
                          7E  7C 0408    713         CLRQ    -(SP)                       ; no status or interrupt routine
                          05  DD 040A    714         PUSHL   #CHC$_DSINT                 ; Disable further interrupts
                          00  DD 040C    715         PUSHL   #0                          ; unit 0
                00000000'EF   04 FB 040E    716         CALLS   #4,DS$CHANNEL               ; disable
                          03  BA 0415    717         POPR    #^M<R0,R1>                  ; Restore
                          02  0417    718         REI                                 ; All status is in L_STATUS(FP)
                              0418    719
```

```
                                0418    721 CON_UBA_RL:
                                0418    722 ;+
                                0418    723 ; now build PT for RL11
                                0418    724 ;-
                                0418    725
                          0B    B0  0418    726         MOVW    #RL_DA_M_STS!RL_DA_M_RST!1,-
                       04 A5        041A    727                 RL_DA(R5)                       ; Setup disk adress register to
                                    041C    728                                                 ; clear errors and get status.
                                    041C    729
                    52    04    D0  041C    730         MOVL    #F_STAT,R2                      ; setup get status function
          52    02    08  0C    AC  F0  041F    731         INSV    12(AP),#8,#2,R2                 ; setup unit # of drive
                    65    52    B0  0425    732         MOVW    R2,RL_CS(R5)                    ; initiate get status to clear errors
                                    0428    733         $DS_WAITUS_S    #1000                   ; Wait for it to finish
                          00    DD  0428                    PUSHL   #0
                 000003E8    8F    DD  C42A                    PUSHL   #1000
              00000000'9F    02    FB  0430                    CALLS   #2, @#DS$WAITUS
                                    0437    734
                 65    0080 8F    B3  0437    735         BITW    #RL_CS_M_CRDY,RL_CS(R5) ; Check if controller ready
                          16    13  043C    736         BEQL    5$                              ; Branch if not ready, Throw it away
                                    043E    737
                    52    40 8F    88  043E    738         BISB2   #RL_CS_M_IE,R2                  ; Setup interrupt enable
                       65    52    B0  0442    739         MOVW    R2,RL_CS(R5)                    ; Now do get status to cause interrupt
                                    0445    740
                                    0445    741         $DS_WAITUS_S    #1000                   ; Wait for it to finish
                          00    DD  0445                    PUSHL   #0
                 000003E8    8F    DD  0447                    PUSHL   #1000
              00000000'9F    02    FB  044D                    CALLS   #2, @#DS$WAITUS
                    54    65    3C  0454    742 5$:     MOVZWL  RL_CS(R5),R4                    ; Get status reg. content
                 65    0040 8F    AA  0457    743         BICW    #RL_CS_M_IE,RL_CS(R5)   ; Clear interrupt enable
                   08 AE    05    D0  045C    744         MOVL    #CHC$_DSINT,8(SP)               ; Change function to disable
                       63    6E    FA  0460    745         CALLG   (SP),(R3)                       ; Disable interrupts
                          11 50    E9  0463    746         BLBC    R0,10$                          ; Return if error
                                    0466    747         TSTW    R4                              ; Check composite error
                          0E    19  0468    748         BLSS    20$                             ; Error if set
          54    54    01    07    EF  046A    749         EXTZV   #RL_MP_V_TYP,#1,R4,R4   ; Isolate drive type, 0=RL01,1=RL02
                                    046F    750
                    51    37    3C  046F    751         MOVZWL  #RL11$K_LEN,R1                  ; Length of RL11 PT
                                81  10  0472    752         BSBB    Allocate_Ptable                 ; Allocate space for same               [07]
                       04 50    E8  0474    753         BLBS    R0,25$                          ; Branch if succeeded
                                    0477    754 10$:
                                04  0477    755         RET                                     ; Return error
                          FDEF  31  0478    756 20$:    BRW     CON_ERROR_X                     ; Leap to error exit
                                    047B    757 25$:
                08 A2    51    B0  047B    758         MOVW    R1,HP$W_SIZE(R2)                ; Set size of structure
                                    047F    759
                    62    04    D0  047F    760         MOVL    #4,HP$Q_DEVICE(R2)              ; Set length of "_DLA"
                 50    0C A2    9E  0482    761         MOVAB   HP$T_DEVICE(R2),R0              ; Set address of "_DLA"
                 04 A2    50    D0  0486    762         MOVL    R0,HP$Q_DEVICE+4(R2)            ; Set address of device name
           60    414C445F 8F    D0  048A    763         MOVL    #^A"_DLA",(R0)                  ; Set "_DLA"
                 18 A2    55    D0  0491    764         MOVL    R5,HP$A_DEVICE(R2)              ; Set device address
                       1C A2    D4  0495    765         CLRL    HP$A_DVA(R2)                    ; Clear virtural address
              20 A2    F4 AD    D0  0498    766         MOVL    L_LINK(FP),HP$A_LINK(R2); Set link to channel
           26 A2    5204 8F    B0  049D    767         MOVW    #^X5204,HP$T_TYPE(R2)   ; Set length and "R" of RL11
        28 A2    0031314C 8F    D0  04A3    768         MOVL    #^A"L11",HP$T_TYPE+2(R2) ; Set "L11"
     32 A2    18 A2    12    00    EF  04AB    769         EXTZV   #0,#18, HP$A_DEVICE(R2), -
                                    04B2    770                 RL11$L_CSR(R2)                  ; Set CSR address
                                    04B2    771
```

```
                24 A2    FE AD    B0  04B2   772          MOVW    L_STATUS+6(FP), -
                                     04B7   773                  HP$W_VECTOR(R2)          ; Set current vector
                      14    13     04B7   774          BEQL    30$                      ; Branch if invalid vector
          50    FC AD    04    02   EF  04B9   775          EXTZV   #CHI$V_IPL,#4, -
                                     04BF   776                  L_STATUS+4(FP),R0        ; Extract BR level of device
                   36 A2    50     90  04BF   777          MOVB    R0,RL11$B_BR(R2)         ; Save BR level
                                     04C3   778
             00000000'EF    62     FA  04C3   779          CALLG   (R2),L^INS$PTABLE        ; Add this PT                          [05]
                      03 50    E8     04CA   780          BLBS    R0,40$                   ; Branch if success
                        FD94   31     04CD   781  30$:     BRW     CON_RELEAS._X            ; Release buffer and exit
                                     04D0   762  40$:
                   F4 AD    52     D0  04D0   783          MOVL    R2,L_LINK(FP)            ; Save PT of RL11
                      51    32     3C  04D4   784          MOVZWL  #RL0T$K_LEN,R1           ; Length of RL01/RL02 PT
                         FF1B   30     04D7   785          BSBW    Allocate_ptable          ; Allocate space for PT                [07]
                      9A 50    E9     C4DA   786          BLBC    R0,10$                   ; Take error exit
                   08 A2    51     B0  04DD   787          MOVW    R1,HP$W_SIZE(R2)         ; Set size of structure
                                     04E1   788
                      62    05     D0  04E1   789          MOVL    #5,HP$Q_DEVICE(R2)       ; Insert length of "_DLA0"
                   50    0C A2     9E  04E4   790          MOVAB   HP$T_DEVICE(R2),R0       ; Insert address of "_DLA0"
                   04 A2    50     D0  04E8   791          MOVL    R0,HP$Q_DEVICE+4(R2)     ; Set address of device name
          80    414C445F  8F     D0  04EC   792          MOVL    #^A"_DLA",(R0)+          ; Insert "_DLA"
           80    0C AC    30     81  04F3   793          ADDB3   #^A"0",12(AP),(R0)+      ; Insert unit number
             0B A2    0C AC    90  04F8   794          MOVB    12(AP),HP$B_DRIVE(R2)    ; Set drive number
                      18 A2    D4  04FD   795          CLRL    HP$A_DEVICE(R2)          ; Clear device address
                      1C A2    D4  0500   796          CLRL    HP$A_DVA(R2)             ; Clear virtural address
                20 A2    F4 AD    D0  0503   797          MOVL    L_LINK(FP),HP$A_LINK(R2) ; Link to RL11
          26 A2    304C5204  8F     D0  0508   798          MOVL    #2^A"@RL0"-60>, -
                                     0510   799                  HP$T_TYPE(R2)            ; Set length=4 and "RL0"
             2A A2    54    31     81  0510   800          ADDB3   #^A"T",R4,HP$T_TYPE+4(R2); Set "1" or "2" if R4=1
                         FEAC   31     0515   801          BRW     CON_NORMAL_X             ; now set default load device
                                     0518   802
```

```
                                0518   804 .sbttl  configure RB730
                                0518   805 ;
                                0518   806 ; Device interrupt handler for RB730
                                0518   807 ;
                                0518   808 ; [warning:  this code really should not expect R5 to be valid.  It
                                0518   809 ;  works because DS$WAITUS doesn't trash R5; however, there is no
                                0518   810 ;  guarantee on this in the future.  THIS SHOULD BE CHANGED IN A FUTURE
                                0518   811 ;  RELEASE, but I am making these alterations 3 working days prior to
                                0518   812 ;  release cut-off, and I just am not up to fancies.  If it works, I'll
                                0518   813 ;  be ecstatic.]
                                0518   814 ;
                                0518   815 RB730_Interrupt:                                ;                                [06]
                          03    BB    0518   816         PushR   #^M<R0,R1>                ; Save volatile regs             [06]
                          7E    7C    051A   817         ClrQ    -(Sp)                     ; no status or interrupt routine [06]
                          05    DD    051C   818         PushL   #CHC$_DSInt               ; Disable further interrupts     [06]
                          00    DD    051E   819         PushL   #0                        ; unit 0                         [06]
          00000000'EF     04    FB    0520   820         CallS   #4, DS$Channel            ; disable                        [06]
          65  00000040   8F    CA    0527   821         Bicl2   #Rb_Cs_M_Ie, Rb_Cs(R5)     ; Disable device interrupts      [06]
                          03    BA    052E   822         PopR    #^M<R0,R1>                ; Restore                        [06]
                          02    0530   823         Rei                               ; All status is in L_STATUS(FP)  [06]
                                0531   824
                                0531   825 con_uba_rb:
                                0531   826
                                0531   827 ;
                                0531   828 ; Build Ptable for RB730 (Nebula IDC)
                                0531   829 ;
          55    40F26200'EF     9E    0531   830         MovAB   ^X40F26200, R5            ; Get proper address for IDC     [08]
                          52    D4    0538   831         clrl    r2                        ; r2 will be shifted unit number
    52    02    08    0C AC     F0    053A   832         insv    12(ap), #8, #2, r2        ; set unit number in right field
                                0540   833
                                0540   834 ;
                                0540   835 ; reset drive, with interrupt on completion
                                0540   836 ;
              10 A5     0B    D0    0540   837         movl    #rb_mp_m_sts-             ; put get status
                                0544   838                  !rb_mp_m_rst-             ;  and reset
                                0544   839                  !rb_mp_m_mrk, -           ;  and mark bit
                                0544   840                  rb_mp(r5)                 ;  into register
    65  00000044 8F     52    C9    0544   841         bis!3   r2, -                     ; put unit number
                                054C   842                  #f_getstatus -            ;  and command
                                054C   843                  !rb_cs_m_ie, -            ;  and interrupt enable bit
                                054C   844                  rb_cs(r5)                 ;  into cs register
                                054C   845         $ds_waitus_s    #1000             ; wait for it to finish
                          00    DD    054C                PUSHL   #0
                000003E8  8F    DD    054E                PUSHL   #1000
          00000000'9F     02    FB    0554                CALLS   #2, @#DS$WAITUS
                54        65    D0    0558   846         movl    rb_cs(r5), r4             ; fetch status register
          65    00000040  8F    CA    055E   847         bicl2   #rb_cs_m_ie, rb_cs(r5)    ; clear interrupt enable
                  08 AE     05    D0    0565   848         movl    #chc$_dsint, 8(sp)        ; change CHANNEL func to disable
                     63     6E    FA    0569   849         callg   (sp), (r3)                ; disable interrupts.
                        12  50    E9    056C   850         blbc    r0, 10$                   ; return if error
                            54    B5    056F   851         tstw    r4                        ; is good status?
                            0F    19    0571   852         blss    20$                       ; composite error set
    54    54      01        1A    EF    0573   853         extzv   #rb_cs_v_typ, #1, r4, r4  ; get type bit (0 = rl02, 1 = rl80)
                            51    37    3C    0578   854         movzwl  #rb730$k_len, r1          ; length of RB730 Ptable
                               +E77   30    057B   855         bsbw    Allocate_ptable           ; allocate space for it          [07]
                              04 50   E8    057E   856         blbs    r0, 30$                   ; check status
                                0581   857 10$:
```

```
                                 04  0581   858          ret                                 ; return error
                                     0582   859 20$:
                           FCE5  31  0582   860          brw     con_error_x                 ; error exit (create status)
                                     0585   861 30$:
                                     0585   862 ;
                                     0585   863 ; device responded: make Ptable
                                     0585   864 ;
                                     0585   865 ; Register usage:
                                     0585   866 ;
                                     0585   867 ;       R1 = size of Ptable
                                     0585   868 ;       R2 = pointer to Ptable
                                     0585   869 ;       R3 = address of DSX$CHANNEL routine
                                     0585   870 ;       R4 = 0 if device is RL02, 1 if R80
                                     0585   871 ;       R5 = address of device CS
                                     0585   872 ;
                                     0585   873
                  08 A2    51   B0  0585   874          movw    r1, hp$w_size(r2)           ; Set size of Ptable
                     62    04   D0  0589   875          movl    #4, hp$q_device(r2)         ; set length of "_DQA"
                  50    0C A2   9E  058C   876          movab   hp$t_device(r2), r0         ; set address of "_DQA"
                  04 A2    50   D0  0590   877          movl    r0, hp$q_device+4(r2)       ; set address of device name
              60  4151445F 8F   D0  0594   878          movl    #^a"_DQA", (r0)             ; set "_DQA" as name
                  18 A2    55   D0  059B   879          movl    r5, hp$a_device(r2)         ; set device address
                     1C A2    D4  059F   880          clrl    hp$a_dva(r2)                ; clear virtual address
                20 A2   F4 AD   D0  05A2   881          movl    l_link(fp), -
                                     05A7   882                  hp$a_link(r2)           ; set link field
                26 A2    5205 8F   B0  05A7   883          movw    #^x5205, hp$t_type(r2)      ; set length and 'R' of RB730
             28 A2   30333742 8F   D0  05AD   884          movl    #^a'B730', -
                                     05B5   885                  hp$t_type+2(r2)         ; set rest of type field
                24 A2   FE AD   B0  05B5   886          movw    l_status+6(fp), -
                                     05BA   887                  hp$w_vector(r2)         ; set vector it interrupted thru
                                12   13  05BA   888          beql    40$                         ; branch if invalid
          50    FC AD   04   02   EF  05BC   889          extzv   #chi$v_ipl, #4, -
                                     05C2   890                  l_status+4(fp), r0      ; get BR level
                   36 A2   50   90  05C2   891          movb    r0, rb730$b_br(r2)          ; save BR level
                0000'CF    62   FA  05C6   892          callg   (r2), w^ins$ptable          ; add this Ptable to list
                        03 50   E8  05CB   893          blbs    r0, 50$                     ; branch if OK
                                     05CE   894 40$:
                           FC93  31  05CE   895          brw     con_release_x               ; release buffer and exit
                                     05D1   896 50$:
                   F4 AD    52   D0  05D1   897          movl    r2, l_link(fp)              ; save link of RB730
                      51    32   3C  05D5   898          movzwl  #rl02$k_len, r1             ; length of RL02/R80 Ptable
                         FE1A  30  05D8   899          bsbw    Allocate_Ptable             ; allocate Ptable                       [07]
                      A3 50   E9  05DB   900          blbc    r0, 10$                     ; exit if error
                   08 A2    51   B0  05DE   901          movw    r1, hp$w_size(r2)           ; set size
                      62    05   D0  05E2   902          movl    #5, hp$q_device(r2)         ; set length of "_DQA0"
                   50    0C A2   9E  05E5   903          movab   hp$t_device(r2), r0         ; address of "_DQA0"
                   04 A2    50   D0  05E9   904          movl    r0, hp$q_device+4(r2)       ; set into descriptor
             80    4151445F 8F   D0  05ED   905          movl    #^a"_DQA", (r0)+            ; set first part of name
              60    0C AC   30   81  05F4   906          addb3   #^a'0', 12(ap), (r0)       ; insert correct unit number
                08 A2    0C AC   90  05F9   907          movb    12(ap), hp$b_drive(r2)      ; set binary unit number
                         18 A2   D4  05FE   908          clrl    hp$a_device(r2)             ; no address
                         1C A2   D4  0601   909          clrl    hp$a_dva(r2)                ; no pass-thru address
                20 A2   F4 AD   D0  0604   910          movl    l_link(fp), -
                                     0609   911                  hp$a_link(r2)           ; set link address to RB730
                   50   26 A2   9E  0609   912          movab   hp$t_type(r2), r0           ; set address of type field
                         09 54   E9  060D   913          blbc    r4, 60$                     ; branch if RL02
                                     0610   914 ;
```

```
                        0610   915 ; Set device type to R80
                        0610   916 ;
      60  30385203 8F   D0  0610   917          movl    #3+<^a'R80'@8>, (r0)    ; set type to counted 'R80'
                  0A    11  0617   918          brb     70$                     ; join common code
                        0619   919 ;
                        0619   920 ; Set device type to RL02
                        0619   921 ;
      80  304C5204 8F   D0  0619   922 60$:     movl    #4+<^a'RL0'@8>, (r0)+   ; set type to counted 'RL02'
          60    32    90  0620   923          movb    #^a'2', (r0)            ;
                        0623   924 ;
                        0623   925 ; Insert Ptable, return
                        0623   926 ;
                  FD9E  31  0623   927 70$:     brw     con_normal_x            ; set load device, exit OK
                        0626   928
```

```
                                0626   930          .Subtitle       Attach UDA-50/LESI load path
                                0626   931
                                0626   932 ;                                                                +       [07]
                                0626   933 ; Since there seems to be no particular purpose to using interupts
                                0626   934 ; when initializing the device, this code uses the perfectly good
                                0626   935 ; PUBTDRIVR code for the purpose, and merely checks the status return.
                                0626   936 ;
                                0626   937
                                0626   938 con_uba_uda:
                                0626   939
                                0626   940 ;
                                0626   941 ; First, turn off interrupts (which were set up in common
                                0626   942 ; UBA code)
                                0626   943 ;
                                C626   944
          08 AE    05    D0     0626   945          MovL    #Chc$_DsInt, 8(Sp)      ; change CHANNEL func to disable
             63    6E    FA     062A   946          CallG   (Sp), (R3)              ; disable interrupts.
                45 50    E9     062D   947          BlbC    R0, 10$                 ; return if error
                                0630   948
                                0630   949 ;
                                0630   950 ; Now, build a pseudo-RPB to make the bootdriver init code
                                0630   951 ; happy, and then call said routine.  After it completes, the
                                0630   952 ; global response buffer is accessible: it has the drive type, etc.
                                0630   953 ;
                                0630   954
                      00    DD  0630   955          PushL   #0                      ; No slave number                       [07]
                   0C AC    DD  0632   956          PushL   12(Ap)                  ; Unit number                           [07]
                50    F4 BD    DE  0635   957          MovAL   @L_Link(Fp), R0         ; Get address of link Ptable            [07]
                   18 A0    DD  0639   958          PushL   Hp$A_Device(R0)         ; Address of link device                [07]
                      55    DD  063C   959          PushL   R5                      ; Address of device CSR                 [07]
                      7E    7C  063E   960          ClrQ    -(Sp)                   ; Don't bother with adap./dev. codes    [07]
00000000'EF           06    FB  0640   961          CallS   #6, Dsr$Allocate_Pseudo_RPB      ; Allocate the RPB             [07]
                      50    D5  0647   962          TstL    R0                      ; If R0 is 0,                           [07]
                      2B    13  0649   963          BEql    20$                     ; .. there was an error                 [07]
                59    50    D0  064B   964          MovL    R0, R9                  ; Put RPB where UD_INIT wants it        [07]
00000000'EF           00    FB  064E   965          CallS   #0, UD_Init             ; Call init routine                     [07]
                   1D 50    E9  0655   966          BlbC    R0, 10$                 ; Return if error                       [07]
                      69    9F  0658   967          PushAB  (R9)                    ; Address of RPB                        [07]
00000000'EF           01    FB  065A   968          CallS   #1, Dsr$DeAllocate_Pseudo_RPB   ; Deallocate the RPB          [07]
59   00000000'EF      9E  0661   969          MovAB   UDA_Response, R9        ; Get address of response buffer        [07]
             59   1C A9    D0  0668   970          MovL    MSCP$L_Media_ID(R9), R9 ; Replace R9 with media ID codes        [07]
                                066C   971
                                066C   972 ;
                                066C   973 ; Build Ptable for UDA-50 (UNIBUS => SDI adapter)
                                066C   974 ; Build Ptable for LESI                                                       [19]
                                066C   975 ;
                                066C   976
                51    38    3C  066C   977          MovZWL  #Hp$K_UDA50_Len, r1     ; length of UDA-50 Ptable
                   FD83    30  066F   978          BsbW    Allocate_Ptable         ; allocate space for it                 [07]
                   04 50    E8  0672   979          BlbS    R0, 30$                 ; check status
                                0675   980 10$:
                                0675   981          Ret                             ; return error
                                0676   982 20$:
                   FBF1    31  0676   983          BrW     Con_Error_X             ; error exit (create status)
                                0679   984 30$:
                                0679   985 ;
                                0679   986 ; device responded: make Ptable
```

```
                                0679   987 ;
                                0679   988 ; Register usage:
                                0679   989 ;
                                0679   990 ;       R1 = size of Ptable
                                0679   991 ;       R2 = pointer to Ptable
                                0679   992 ;       R3 = scratch
                                0679   993 ;       R4 = scratch
                                0679   994 ;       R5 = address of device CS
                                0679   995 ;       R9 = UDA-50 drive media ID code
                                0679   996 ;
                                0679   997 ;               31   26   21   16   11  7 6    0
                                0679   998 ;               +----+----+----+----+----+------+
                                0679   999 ;               | D0 | D1 | A0 | A1 | A2 |   N  |
                                0679  1000 ;               +----+----+----+----+----+------+
                                0679  1001 ;
                                0679  1002 ;                       D0, D1 : 5-bit ASCII preferred device
                                0679  1003 ;                                generic prefix for unit. 1='A', 2='B'.
                                0679  1004 ;                                D0 is left letter, D1 is right.
                                0679  1005 ;                       A0,A1,A2,N : Media name of unit.  Up to 3 alpha
                                0679  1006 ;                                characters (left-justified in A0-2), and
                                0679  1007 ;                                N (two decimal digits).  Ex: A0=17,
                                0679  1008 ;                                A1=1, A2=0, N=80; name is 'RA80'.
                                0679  1009 ;                                Name can also be RA81,RA60,RC25,RCF25
                                0679  1010 ;
                                0679  1011
          08 A2    51    B0   0679  1012          MovW    R1, Hp$W_Size(R2)       ; Set size of Ptable
             62    04    D0   067D  1013          MovL    #4, Hp$Q_Device(R2)     ; set length of "_DUA"
          50    0C A2    9E   0680  1014          MovAB   Hp$T_Device(R2), R0     ; set address of "_DUA"
          04 A2    50    D0   0684  1015          MovL    R0, Hp$Q_Device+4(R2)   ; set address of device name
    60  4155445F    8F    D0   0688  1016          MovL    #^A"_DUA", (R0)         ; set "_DUA" as name/DEFAULT                [10]
          18 A2    55    D0   068F  1017          MovL    R5, Hp$A_Device(R2)     ; set device address
32 A2    55    12    00    EF   0693  1018          ExtZV   #0, #18, R5, -          ; Set UNIBUS address into
                                0699  1019                   Hp$L_Uda50_UdaIp(R2) ; .. correct field of Ptable
             1C A2    D4   0699  1020          ClrL    Hp$A_Dva(R2)            ; clear sibling address
          20 A2    F4 AD    D0   069C  1021          MovL    L_Link(Fp), -
                                06A1  1022                   Hp$A_Link(R2)       ; set link field
    20643019 8F    59    D1   06A1  1023          CMPL    R9,#^X20643019          ; Check for RC25                            [10]
                   09    13   06A8  1024          BEQL    33$                     ; Branch if found.                          [10]
    20643319 8F    59    D1   06AA  1025          CMPL    R9,#^X20643319          ; Check for RCF25                           [10]
                   18    12   06B1  1026          BNEQ    35$                     ; Branch if found.                          [10]
                                06B3  1027
     26 A2   4C04 8F    B0   06B3  1028 33$:     MOVW    #4!<^A"L"@8>, -         ; Set length and 'L'                        [13]
                                06B9  1029                   HP$T_TYPE(R2)       ; of LESI                                   [10]
 28 A2    20495345 8F    D0   06B9  1030          MOVL    #^A"ESI ", -            ; and set                                   [13]
                                06C1  1031                   HP$T_TYPE+2(R2)     ; rest of type field                        [10]
 0C A2    4141445F 8F    D0   06C1  1032          MOVL    #^A"_DAA",HP$T_DEVICE(R2); RESET "_DAA" AS NAME                     [18]
                   0E    11   06C9  1033          BRB     38$                     ; Branch to continue                        [10]
                                06CB  1034
     26 A2   5505 8F    B0   06CB  1035 35$:     MovW    #5!<^A"U"@8>, -         ; Set length and 'U'
                                06D1  1036                   Hp$T_Type(R2)       ; .. of UDA50
 28 A2    30354144 8F    D0   06D1  1037          MovL    #^A"DA50", -            ; And set
                                06D9  1038                   Hp$T_Type+2(R2)     ; .. rest of type field
                                06D9  1039
     24 A2   006C 8F    B0   06D9  1040 38$:     MovW    #^0154, Hp$W_Vector(R2) ; Set vector of fst UDA
          36 A2    05    90   06DF  1041          MovB    #5, Hp$B_UDA50_Br(R2)   ; save BR level
          37 A2    02    90   06E3  1042          MovB    #2, Hp$B_UDA50_Burst(R2); Assume default burst rate                 [07]
       0000'CF    62    FA   06E7  1043          CallG   (R2), W^Ins$Ptable      ; add this Ptable to list
```

```
                03 50    E8  06EC  1044            BlbS    R0, 50$                 ; branch if OK
                                06EF  1045 40$:
                 FB72    31  06EF  1046            BrW     Con_Release_X           ; release buffer and exit
                                06F2  1047 50$:
          F4 AD    52    D0  06F2  1048            MovL    R2, L_Link(Fp)          ; save link of UDA50/LESI
             51    32    3C  06F6  1C49            MovZWL  #RA80$K_Len, R1         ; length of RA80/81/60,RC25,RCF25 Ptable
               FCF9    30  06F9  1050            BsbW    Allocate_Ptable         ; allocate Ptable                       [07]
             03 50    E8  06FC  1051            BlbS    R0, 52$                 ; continue if no error                  [10]
               FF73    31  06FF  1052            BRW     10$                     ; exit if error                         [10]
          08 A2    51    B0  0702  1053 52$:     MovW    R1, Hp$W_Size(R2)       ; set size
             62    05    D0  0706  1054            MovL    #5, Hp$Q_Device(R2)     ; set length of "_DUA0"
          50    0C A2    9E  0709  1055            MovAB   Hp$T_Device(R2), R0     ; address of "_DUA0"
          04 A2    50    D0  070D  1056            MovL    R0, Hp$Q_Device+4(R2)   ; set into descriptor
    60  4155445F  8F    D0  0711  1057            MovL    #^A"_DUA", (R0)         ; set first part of name,DEFAULT        [10]
 53   59    0A    16  EF  0718  1058            ExtZV   #22, #10, R9, R3        ; Get preferred name
                16    13  071D  1059            BEql    55$                     ; Branch if null
 54   53    05    00  EF  071F  1060            ExtZV   #0, #5, R3, R4          ; Get second character
 53   53    05    05  EF  0724  1061            ExtZV   #5, #5, R3, R3          ; Get first character
01 A0   53    40  8F  81  0729  1062            AddB3   #<^A"A"-1>, R3, 1(R0)   ; Replace first character
02 A0   54    40  8F  81  072F  1063            AddB3   #<^A"A"-1>, R4, 2(R0)   ; .. and second as well
             50    04  C0  0735  1064 55$:     AddL2   #4, R0                  ; Increment pointer
    60    0C AC    30  81  0738  1065            AddB3   #^A"0", 12(Ap), (R0)    ; insert correct unit number
       0B A2    0C AC    90  073D  1066            MovB    12(Ap), Hp$B_Drive(R2)  ; set binary unit number
                18 A2    D4  0742  1067            ClrL    Hp$A_Device(R2)         ; no address
                1C A2    D4  0745  1068            ClrL    Hp$A_Dva(R2)            ; no pass-thru address
          20 A2    F4 AD    D0  0748  1069            MovL    L_Link(Fp), -           ;
                                074D  1070                     Hp$A_Link(R2)       ; set link address to UDA50/LESI        [10]
             50    26 A2    9E  074D  1071            MovAB   Hp$T_Type(R2), R0       ; set address of type field
                   50    D6  0751  1072            IncL    R0                      ; Skip count byte
                51    50    D0  0753  1073            MovL    R0, R1                  ; Copy start of length
                53    11    D0  0756  1074            MovL    #17, R3                 ; Bit of first character
       54   59    05    53  EF  0759  1075 60$:     ExtZV   R3, #5, R9, R4          ; Get next character
                   0F    13  075E  1076            BEql    65$                     ; Exit if null
          80    54    40  8F  81  0760  1077            AddB3   #<^A"A"-1>, R4, (R0)+   ; Otherwise, store it
FFEA 53  FFFFFFFB  8F    06  F1  0765  1078            AcbL    #6, #-5, R3, 60$        ; Loop through characters
       53   59    07    00  EF  076F  1079 65$:     ExtZV   #0, #7, R9, R3          ; Get number
                   54    D4  0774  1080            ClrL    R4                      ; Clear high order
       54   53    53    0A  7B  0776  1081            EDiv    #10, R3, R3, R4         ; Divide by 10
            80    53    30  81  077B  1082            AddB3   #^A"0", R3, (R0)+       ; First digit
            80    54    30  81  077F  1083            AddB3   #^A"0", R4, (R0)+       ; Second digit
                  50    51  C2  0783  1084            SubL2   R1, R0                  ; Get length
                  71    50  90  0786  1085            MovB    R0, -(R1)               ; Store byte length
                              0789  1086
                              0789  1087 ;
                              0789  1088 ; Insert Ptable, return
                              0789  1089 ;
                              0789  1090
                 FC38    31  0789  1091 70$:     BrW     Con_Normal_X            ; set load device, exit OK
                              078C  1092                                          ;                               -       [07]
                              078C  1093
                              078C  1094            .END
```

```
$ER                             = 00000003     D
$MODULE                           00000000 R   D   02
ALLOCATE_PTABLE                   000003F5 R   D   03
BOOT$L_ADP                        ********     X   00
BOOT$L_CSR                        ********     X   00
BOOT$L_DEVTYP                     ********     X   00
BOOT$L_R2                         ********     X   00
BOOT$L_UNIT                       ********     X   00
BTD$K_HSCCI                     = 00000020     D
BTD$K_UDA                       = 00000011     D
CHC$_ABORT                      = 00000002     D
CHC$_CLEAR                      = 00000006     D
CHC$_CLRDFT                     = 00000009     D
CHC$_DSINT                      = 00000005     D
CHC$_ENINT                      = 00000004     D
CHC$_INITA                      = 00000000     D
CHC$_INITB                      = 00000001     D
CHC$_PURGE                      = 00000003     D
CHC$_SELF_TEST                  = 0000000B     D
CHC$_SETDFT                     = 00000008     D
CHC$_STATUS                     = 00000007     D
CHC$_STOP                       = 0000000A     D
CHI$M_CHNINT                    = 00000001     D
CHI$M_DEVINT                    = 00000002     D
CHI$M_IPL                       = 0000007C     D
CHI$S_IPL                       = 00000005     D
CHI$V_CHNINT                    = 00000000     D
CHI$V_DEVINT                    = 00000001     D
CHI$V_IPL                       = 00000002     D
CI$B_BR                           00000033     D
CI$B_NODE                         00000034     D
CI$B_TR                           00000032     D
CI$K_LEN                          00000041     D
CI$L_FUNC                         00000035     D
CI$L_INDEX                        0000003D     D
CI$L_MAINT_ID                     00000039     D
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        00000444     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEST                        00000020     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
COMMON_INTERRUPT                  00000406 R   D   03
CON_CI_HSCCI                      000000D0 R   D   03
CON_ERROR_X                       0000026A R   D   03
CON_MBA                           000001E2 R   D   03
CON_NORMAL_X                      000003C4 R   D   03
CON_RELEASE_X                     00000264 R   D   03
CON_UBA                           00000272 R   D   03
CON_UBA_RB                        00000531 R   D   03
CON_UBA_RK                        000002E4 R   D   03
CON_UBA_RL                        00000418 R   D   03
CON_UBA_UDA                       00000626 R   D   03
CON_X                             00000271 R   D   03
DIR...                          = FFFFFFFF     D
DISK$K_LEN                        00000032     D
DS$CHANNEL                        ********     X   00
DS$GA_SELECTED                    ********     X   00
DS$GL_CLIBASE                     ********     X   00
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$WAITUS                         ********     X   00
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
```

```
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
DS$_IHWE                        = 00660060    D
DS$_ILLCHAR                     = 00660018    D
DS$_ILLPAGCNT                   = 00660078    D
DS$_ILLUNIT                     = 00660100    D
DS$_INSFMEM                     = 00660050    D
DS$_IPL2HI                      = 006600B8    D
DS$_IVADDR                      = 00660040    D
DS$_IVVECT                      = 00660038    D
DS$_KRNLSTK                     = 00660090    D
DS$_LOGIC                       = 00660070    D
DS$_MCHK                        = 00660088    D
DS$_MMOFF                       = 00660058    D
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSP$CONFIG_ADP                  ********    X   00
DSP$CONFIG_LOAD                 0000007F RG D   03
DSR$ALLOCATE_PSEUDO_RPB         ********    X   00
DSR$DEALLOCATE_PSEUDO_RPB       ********    X   00
DSV$SETLOAD                     ********    X   00
DS_ERRSUP                       ********    X   00
DT_RM03                         = 000C0015    D
DT_RM05                         = 00000017    D
DT_RM80                         = 00000016    D
DT_RP04                         = 00000010    D
DT_RP05                         = 00000011    D
DT_RP06                         = 00000012    D
DT_RP07                         = 00000022    D
DT_RP07_HPT                     = 00000021    D
EXE$ALONONPAGED                 ********    X   00
EXE$DEANONPAGED                 ********    X   00
F_DRVCLR                        = 00000004    D
F_GETSTATUS                     = 00000004    D
F_NOP                           = 00000000    D
F_STAT                          = 00000004    D
HP$A_DEPENDENT                    00000032    D
HP$A_DEVICE                       00000018    D
HP$A_DVA                          0000001C    D
HP$A_LINK                         00000020    D
HP$B_CI_BR                        00000033    D
HP$B_CI_NODE                      00000034    D
HP$B_CI_TR                        00000032    D
HP$B_DRIVE                        00000008    D
HP$B_FLAGS                        0000000A    D
HP$B_RB730_BR                     00000036    D
HP$B_RK611_BR                     00000036    D
HP$B_RL11_BR                      00000036    D
HP$B_UDA50_BR                     00000036    D
HP$B_UDA50_BURST                  00000037    D
HP$K_CI_LEN                       00000041    D
HP$K_DISK_LEN                     00000032    D
HP$K_RA80_LEN                     00000032    D
HP$K_RB730_LEN                    00000037    D
HP$K_RK06_LEN                     00000032    D
HP$K_RK611_LEN                    00000037    D
HP$K_RL01_LEN                     00000032    D
HP$K_RL02_LEN                     00000032    D
HP$K_RL11_LEN                     00000037    D
HP$K_RP04_LEN                     00000032    D
HP$K_UDA50_LEN                    00000038    D
HP$L_CI_FUNC                      00000035    D
HP$L_CI_INDEX                     0000003D    D
HP$L_CI_MAINT_ID                  00000039    D
HP$L_RB730_CSR                    00000032    D
HP$L_RK611_CSR                    00000032    D
HP$L_RL11_CSR                     00000032    D
HP$L_UDA50_UDAIP                  00000032    D
HP$Q_DEVICE                       00000000    D
HP$T_DEVICE                       0000000C    D
HP$T_TYPE                         00000026    D
HP$W_SIZE                         00000008    D
HP$W_VECTOR                       00000024    D
INI$LOAD_DEVICE                   00000000 RG D   03
INS$PTABLE                      ********    X   00
IOC$K_IOSPACE                   ********    X   00
K_TYPES                         = 00000013    D
```

```
L_LINK                          FFFFFFF4    D
L_LOCAL                         FFFFFFF0    D
L_STATUS                        FFFFFFF8    D
L_TYPE                          FFFFFFF0    D
MSCP$L_MEDIA_ID               = 0000001C    D
R80$K_LEN                       00000032    D
RA80$K_LEN                      00000032    D
RB730$B_BR                      00000036    D
RB730$K_LEN                     00000037    D
RB730$L_CSR                     00000032    D
RB730_INTERRUPT                 00000518 R  D  03
RB_BA                           00000004    D
RB_BC                           00000008    D
RB_CS                           00000000    D
RB_CS_M_IE                    = 00000040    D
RB_CS_V_TYP                   = 0000001A    D
RB_DA                           0000000C    D
RB_MP                           00000010    D
RB_MP_M_MRK                   = 00000001    D
RB_MP_M_RST                   = 00000008    D
RB_MP_M_STS                   = 00000002    D
RK06$K_LEN                      00000032    D
RK611$B_BR                      00000036    D
RK611$K_LEN                     00000037    D
RK611$L_CSR                     00000032    D
RK_CS1                        = 00000000    D
RK_CS1_M_CERR                 = 00008000    D
RK_CS1_M_IE                   = 00000040    D
RK_CS1_M_RDY                  = 00000080    D
RK_CS2                        = 00000008    D
RK_DS                         = 0000000A    D
RK_DS_V_DDT                   = 00000008    D
RL01$K_LEN                      00000032    D
RL02$K_LEN                      00000032    D
RL11$B_BR                       00000036    D
RL11$K_LEN                      00000037    D
RL11$L_CSR                      00000032    D
RL_CS                         = 00000000    D
RL_CS_M_CRDY                  = 00000080    D
RL_CS_M_IE                    = 00000040    D
RL_DA                         = 00000004    D
RL_DA_M_MRK                   = 00000001    D
RL_DA_M_RST                   = 00000008    D
RL_DA_M_STS                   = 00000002    D
RL_MP                         = 00000006    D
RL_MP_V_TYP                   = 00000007    D
RP04$K_LEN                      00000032    D
RP_CS1                        = 00000000    D
RP_DT                         = 00000018    D
RP_DT_S_DTN                   = 00000009    D
RP_DT_V_DTN                   = 00000000    D
SAVABS...                     = FFFFFFF0    D
SCB_BASE                        ********    X    0C
SCB_IMAGE                       ********    X    00
SIZ...                        = 00000002    D
SS$_NORMAL                    = 00000001    D
SYS$UNWIND                      ********    X    00

T_TYPES                         00000007 R  D    02
UDA50$B_BR                      00000036    D
UDA50$B_BURST                   00000037    D
UDA50$K_LEN                     00000038    D
UDA50$L_UDAIP                   00000032    D
UDA_RESPONSE                    ********    X    00
UD_INIT                         ********    X    00
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFF8 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	0000009F (159.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	0000078C (1932.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000003	542 (7)	#-442 (5) #-542 (7)
$MODULE	00000000-R	290 (2)	442 (5) 542 (7)
ALLOCATE_PTABLE	000003F5-R	704 (9)	#-1050 (10) #-452 (6) #-487 (6) #-529 (7) #-623 (9) #-656 (9) #-752 (10) #-785 (10) #-855 (10) #-899 (10) #-978 (10)
BIT...	=00000010	150 (2)	146 (2) 149 (2) 150 (2)
BOOT$L_ADP	00000000-XR		286 (2) #-363 (3)
BOOT$L_CSR	00000C00-XR		286 (2) #-368 (3)
BOOT$L_DEVTYP	00000000-XR		286 (2) #-366 (3)
BOOT$L_R2	00000000-XR		286 (2) #-365 (3)
BOOT$L_UNIT	00000000-XR		286 (2) #-367 (3)
BTD$K_HSCCI	=00000020		#-437 (5)
BTD$K_UDA	=00000011		#-434 (5) #-596 (8)
CHC$_ABORT	=00000002	149 (2)	
CHC$_CLEAR	=00000006	149 (2)	#-588 (8)
CHC$_CLRDFT	=00000009	149 (2)	
CHC$_DSINT	=00000005	149 (2)	#-615 (9) #-714 (9) #-744 (10) #-618 (10) #-848 (10) #-945 (10)
CHC$_ENINT	=00000004	149 (2)	#-591 (8)
CHC$_INITA	=00000000	149 (2)	#-581 (8)
CHC$_INITB	=00000001	149 (2)	#-585 (8)
CHC$_PURGE	=00000003	149 (2)	
CHC$_SELF_TEST	=0000000B	149 (2)	
CHC$_SETDFT	=00000008	149 (2)	
CHC$_STATUS	=00000007	149 (2)	
CHC$_STOP	=0000000A	149 (2)	
CHI$M_CHNINT	=00000001	150 (2)	
CHI$M_DEVINT	=00000002	150 (2)	
CHI$M_IPL	=0000007C	150 (2)	
CHI$S_IPL	=00000005	150 (2)	
CHI$V_CHNINT	=00000000	150 (2)	
CHI$V_DEVINT	=00000001	150 (2)	
CHI$V_IPL	=00000002	150 (2)	#-646 (9) #-775 (10) #-889 (10)
CLI$K_BUFSIZ	=00000100	118 (2)	118 (2)
CLI$K_SIZE	00000444	118 (2)	
CLI$L_ADDRESS	00000018	118 (2)	
CLI$L_COMMAND	00000004	118 (2)	
CLI$L_DATA	0000001C	118 (2)	
CLI$L_FLAGS	00000000	118 (2)	
CLI$L_LAST	00000024	118 (2)	
CLI$L_NEXT	00000030	118 (2)	
CLI$L_PASS	0000002C	118 (2)	
CLI$L_SUBT	00000028	118 (2)	
CLI$L_TEST	00000020	118 (2)	
CLI$Q_BUFQWD	00000034	118 (2)	
CLI$Q_FILE	00000008	118 (2)	#-678 (9) #-684 (9)
CLI$Q_SECTION	00000010	118 (2)	
CLI$Q_TIME	0000043C	118 (2)	
CLI$T_BUFFER	0000003C	118 (2)	
CLI$V_ADAPTER	=00000018	118 (2)	

```
CLI$V_ADR                       =0000000B      118     (2)
CLI$V_ASCII                     =00000013      118     (2)
CLI$V_BREAK                     =0000000A      118     (2)
CLI$V_BRIEF                     =0000001B      118     (2)
CLI$V_BYTE                      =0000000D      118     (2)
CLI$V_CLEAR                     =00000002      118     (2)
CLI$V_DEC                       =00000010      118     (2)
CLI$V_DEFAULT                   =0000000C      118     (2)
CLI$V_DEPOSIT                   =00000019      118     (2)
CLI$V_EVENT                     =00000008      118     (2)
CLI$V_EXAM                      =00000005      118     (2)
CLI$V_FLAGS                     =00000009      118     (2)
CLI$V_HEX                       =00000012      118     (2)
CLI$V_KERNEL                    =00000017      118     (2)
CLI$V_LOAD                      =00000C06      118     (2)
CLI$V_LONG                      =0000000F      118     (2)
CLI$V_NOTNUF                    =00000001      118     (2)
CLI$V_OCT                       =00000011      118     (2)
CLI$V_PREG                      =0000001A      118     (2)
CLI$V_QA                        =00000007      118     (2)
CLI$V_QACKLOOPLOOPS             =0000001C      118     (2)
CLI$V_QAERRORPRINTS             =0000001B      118     (2)
CLI$V_QAMULTIPLEPASS            =0000001F      118     (2)
CLI$V_QASUBTESTLOOPS            =0000001E      118     (2)
CLI$V_QATESTLOOPS               =0000001D      118     (2)
CLI$V_REG                       =00000014      118     (2)
CLI$V_REQUIRED                  =00000000      118     (2)
CLI$V_RUN                       =00000015      118     (2)
CLI$V_SET                       =00000003      118     (2)
CLI$V_SHOW                      =00000004      118     (2)
CLI$V_VALSEC                    =00000016      118     (2)
CLI$V_WORD                      =0000000E      118     (2)
COMMON_INTERRUPT                 00000406-R    711     (9)       577     (8)
CON_CI_HSCCI                     000000D0-R    447     (6)     #-439     (5)
CON_ERROR_X                      0000026A-R    563     (7)     #-627     (9)     #-756     (10)    #-860     (10)    #-983     (10)
CON_MBA                          000001E2-R    518     (7)       441     (5)
CON_NORMAL_X                     000003C4-R    672     (9)     #-1091    (10)    #-515     (6)     #-558     (7)     #-801     (10)
                                                               #-927     (10)
CON_RELEASE_X                    00000264-R    560     (7)     #-1046    (10)    #-485     (6)     #-543     (7)     #-652     (9)
                                                               #-781     (10)    #-895     (10)
CON_UBA                          00000272-R    568     (8)     #-436     (5)       441     (5)
CON_UBA_RB                       00000531-R    825     (10)      600     (8)
CON_UBA_RK                       000002E4-R    604     (9)       600     (8)
CON_UBA_RL                       00000418-R    721     (10)      600     (8)
CON_UBA_UDA                      00000626-R    938     (10)    #-598     (8)
CON_X                            00000271-R    565     (7)     #-584     (8)     #-587     (8)     #-590     (8)     #-593     (8)
DIR...                          =FFFFFFFF      422     (4)       422     (4)
DS$CHANNEL                       00000000-XR                    279     (2)       575     (8)       716     (9)       820     (10)
DS$GA_SELECTED                   00000000-XR                    282     (2)     #-573     (8)
DS$GL_CLIBASE                    00000000-XR                    282     (2)       676     (9)
DS$K_ERROR                      =00000002      146     (2)
DS$K_NORMAL                     =00000001      146     (2)
DS$K_SEVERE                     =00000004      146     (2)
DS$K_SUBSYS                     =00000066      146     (2)       146     (2)
DS$K_WARNING                    =00000000      146     (2)
DS$WAITUS                        00000000-XR                    279     (2)       612     (9)       733     (10)      741     (10)
                                                                845     (10)
```

```
DS$_ARITH                       =006600D0      146     (2)
DS$_ASBE                        =00660118      146     (2)
DS$_BADLINK                     =006600F0      146     (2)
DS$_BADTYPE                     =006600E8      146     (2)
DS$_BIIC                        =00660120      146     (2)
DS$_CHME                        =006600A8      146     (2)
DS$_CHMK                        =006600E0      146     (2)
DS$_DEVNAME                     =00660108      146     (2)
DS$_ERROR                       =00660002      146     (2)    #-443     (5)    #-564     (7)
DS$_FHWE                        =00660068      146     (2)
DS$_FRAGBUF                     =00660080      146     (2)
DS$_ICBUSY                      =006600C8      146     (2)
DS$_ICERR                       =006600C0      146     (2)
DS$_IHWE                        =00660060      146     (2)
DS$_ILLCHAR                     =00660C18      146     (2)
DS$_ILLPAGCNT                   =00660078      146     (2)
DS$_ILLUNIT                     =00660100      146     (2)
DS$_INSFMEM                     =00660050      146     (2)
DS$_IPL2HI                      =006600B8      146     (2)
DS$_IVADDR                      =00660040      146     (2)
DS$_IVVECT                      =00660038      146     (2)
DS$_KRNLSTK                     =00660090      146     (2)
DS$_LOGIC                       =00660070      146     (2)
DS$_MCHK                        =00660088      146     (2)
DS$_MMOFF                       =00660058      146     (2)
DS$_NEEDUNIT                    =006600F8      146     (2)
DS$_NODE                        =00660128      146     (2)
DS$_NOPCS                       =00660110      146     (2)
DS$_NORMAL                      =00660001      146     (2)
DS$_NOSUPPORT                   =006600B1      146     (2)
DS$_NOTDON                      =00660030      146     (2)
DS$_NOTIMP                      =006600B0      146     (2)      146     (2)
DS$_NULLSTR                     =00660010      146     (2)
DS$_OVERFLOW                    =00660008      146     (2)
DS$_POWER                       =00660098      146     (2)
DS$_PROGERR                     =00660020      146     (2)
DS$_SEVERE                      =00660004      146     (2)
DS$_TRANSL                      =006600A0      146     (2)
DS$_TRUNCATE                    =00660028      146     (2)
DS$_UNEXPINT                    =006600D8      146     (2)
DS$_VASFULL                     =00660048      146     (2)
DS$_WARNING                     =00660000      146     (2)      146     (2)
DSA$GL_UNITS                     0000FE0C                      #-572     (8)
DSP$CONFIG_ADP                   00000000-XR                     279     (2)      431     (5)
DSP$CONFIG_LOAD                  0000007F-R     425     (5)      370     (3)
DSR$ALLOCATE_PSEUDO_RPB          00000J00-XR                     284     (2)      961     (10)
DSR$DEALLOCATE_PSEUDO_RPB        00000000-XR                     285     (2)      968     (10)
DSV$SETLOAD                      00000000-XR                     230     (2)    #-685     (9)
DS_ERRSUP                        00000000-XR                     279     (2)      442     (5)      542     (7)
DT_RM03                         =00000015      183     (2)
DT_RM05                         =00000017      185     (2)
DT_RM80                         =00000016      184     (2)
DT_RP04                         =00000010      178     (2)    #-536     (7)
DT_RP05                         =00000011      179     (2)
DT_RP06                         =00000012      180     (2)
DT_RP07                         =00000022      182     (2)
DT_RP07_HPT                     =00000021      181     (2)
```

```
EXE$ALONONPAGED                 00000000-XR                      278 (2)      344 (3)      705 (9)
EXE$DEANONPAGED                 00000000-XR                      278 (2)      361 (3)    #-562 (7)
F_DRVCLR                        =00000004      159     (2)     #-610 (9)
F_GETSTATUS                     =00000004      272     (2)     #-842 (10)
F_NOP                           =00000000      174     (2)     #-524 (7)
F_STAT                          =00000004      171     (2)     #-730 (10)
HP$A_DEPENDENT                   00000032                       #-343 (3)
HP$A_DEVICE                      00000018                       #-1017 (10)  #-1067 (10)  #-475 (6)   #-476 (6)
                                                                #-555 (7)    #-635 (9)     640 (9)    #-666 (9)
                                                                #-764 (10)    769 (10)   #-795 (10)   #-879 (10)
                                                                #-908 (10)   #-958 (10)
HP$A_DVA                         0000001C                       #-1020 (10)  #-1068 (10)  #-476 (6)   #-513 (6)
                                                                #-556 (7)    #-636 (9)    #-667 (9)    #-765 (10)
                                                                #-796 (10)   #-880 (10)   #-909 (10)
HP$A_LINK                        00000C20                       #-1022 (10)  #-1070 (10)  #-477 (6)   #-514 (6)
                                                                #-557 (7)    #-637 (9)    #-668 (9)    #-766 (10)
                                                                #-797 (10)   #-882 (10)   #-911 (10)
HP$B_CI_NODE                     00000034                       #-481 (6)
HP$B_DRIVE                       0000000B                       #-1066 (10)  #-474 (6)    #-512 (6)    #-553 (7)
                                                                #-665 (9)    #-794 (10)   #-907 (10)
HP$B_UDA50_BR                    00000036                       #-1041 (10)
HP$B_UDA50_BURST                 00000037                       #-1042 (10)
HP$K_CI_LEN                      00000041                       #-451 (6)
HP$K_DISK_LEN                    00000032                       #-486 (6)
HP$K_UDA50_LEN                   00000038                       #-977 (10)
HP$L_CI_FUNC                     00000035                       #-480 (6)
HP$L_CI_INDEX                    0000003D                       #-478 (6)
HP$L_CI_MAINT_ID                 00000039                       #-479 (6)
HP$L_UDA50_UDAIP                 00000032                       #-1019 (10)
HP$Q_DEVICE                      00000000                       #-1013 (10)  #-1015 (10)  #-1054 (10)  #-1056 (10)
                                                                #-350 (3)    #-352 (3)    #-458 (6)    #-460 (6)
                                                                #-467 (6)    #-472 (6)    #-493 (6)    #-495 (6)
                                                                #-504 (6)    #-510 (6)    #-548 (7)    #-550 (7)
                                                                #-631 (9)    #-633 (9)    #-660 (9)    #-662 (9)
                                                                #-677 (9)    #-680 (9)    #-760 (10)   #-762 (10)
                                                                #-789 (10)   #-791 (10)   #-875 (10)   #-877 (10)
                                                                #-902 (10)   #-904 (10)
HP$T_DEVICE                      0000000C                        1014 (10)   #-1032 (10)   1055 (10)    351 (3)
                                                                 459 (6)      494 (6)      549 (7)      632 (9)
                                                                 661 (9)    #-682 (9)      761 (10)     790 (10)
                                                                 876 (10)     903 (10)
HP$T_TYPE                        00000026                       #-1029 (10)  #-1031 (10)  #-1036 (10)  #-1038 (10)
                                                                1071 (10)     355 (3)      #-456 (6)    #-457 (6)
                                                                #-491 (6)    #-492 (6)    #-545 (7)    #-546 (7)
                                                                #-638 (9)    #-639 (9)    #-670 (9)    #-671 (9)
                                                                #-767 (10)   #-768 (10)   #-799 (10)   #-800 (10)
                                                                #-883 (10)   #-885 (10)    912 (10)
HP$W_SIZE                        00000008                       #-1012 (10)  #-1053 (10)  #-349 (3)   #-455 (6)
                                                                #-490 (6)    #-534 (7)    #-629 (9)    #-658 (9)
                                                                #-758 (10)   #-787 (10)   #-874 (10)   #-901 (10)
HP$W_VECTOR                      00000024                       #-1040 (10)  #-644 (9)    #-773 (10)   #-887 (10)
INI$LOAD_DEVICE                  00000000-R     341     (3)
INS$PTABLE                       00000000-XR                     1043 (10)     279 (2)      358 (3)      483 (6)
                                                                 650 (9)      673 (9)      779 (10)     892 (10)
IOC$K_IOSPACE                    00000000-XR                     277 (2)    #-475 (6)    #-522 (7)    #-594 (8)
K_TYPES                          =00000013      304     (2)     #-538 (7)
L_LINK                           FFFFFFF4       422     (4)     #-1021 (10)  #-1048 (10)  #-1069 (10)   429 (5)
```

```
                                                        #-477    (6)     #-557    (7)     #-573    (8)     #-637    (9)
                                                        #-654    (9)     #-668    (9)     #-766    (10)    #-783    (10)
                                                        #-797    (10)    #-881    (10)    #-897    (10)    #-910    (10)
                                                          957    (10)
L_LOCAL                  FFFFFFF0      422    (4)        427    (5)
L_STATUS                 FFFFFFF8      422    (4)        576    (8)     #-643    (9)      647    (9)     #-772    (10)
                                                          776    (10)    #-886    (10)     890    (10)
L_TYPE                   FFFFFFF0      422    (4)        428    (5)
MSCP$L_MEDIA_ID         =0000001C                        #-970    (10)
RA80$K_LEN               00000032                        #-1049   (10)
RB730$B_BR               00000036                        #-891    (10)
RB730$K_LEN              00000037                        #-854    (10)
RB730_INTERRUPT          00000518-R    815    (10)       580    (8)
RB_CS                    00000000                        #-821    (10)    #-844    (10)    #-846    (10)    #-847    (10)
RB_CS_M_IE              =00000040                        #-821    (10)    #-843    (10)    #-847    (10)
RB_CS_V_TYP             =0000001A                        #-853    (10)
RB_MP                    00000010                        #-840    (10)
RB_MP_M_MRK             =00000001                        #-839    (10)
RB_MP_M_RST             =00000008                        #-839    (10)
RB_MP_M_STS             =00000002                        #-838    (10)
RK06$K_LEN               00000032                        #-655    (9)
RK611$B_BR               00000036                        #-648    (9)
RK611$K_LEN              00000037                        #-622    (9)
RK611$L_CSR              00000032                        #-641    (9)
RK_CS1                  =00000000      152    (2)        #-608    (9)     #-611    (9)     #-614    (9)
RK_CS1_M_CERR           =00008000      155    (2)        #-608    (9)
RK_CS1_M_IE             =00000040      153    (2)        #-610    (9)     #-614    (9)
RK_CS1_M_RDY            =00000080      154    (2)
RK_CS2                  =00000008      156    (2)        #-609    (9)
RK_DS                   =0000000A      157    (2)        #-613    (9)
RK_DS_V_DDT             =00000008      158    (2)        #-620    (9)
RL01$K_LEN               00000032                        #-784    (10)
RL02$K_LEN               00000032                        #-898    (10)
RL11$B_BR                00000036                        #-777    (10)
RL11$K_LEN               00000037                        #-751    (10)
RL11$L_CSR               00000032                        #-770    (10)
RL_CS                   =00000000      162    (2)        #-732    (10)    #-735    (10)    #-739    (10)    #-742    (10)
                                                          #-743    (10)
RL_CS_M_CRDY            =00000080      163    (2)        #-735    (10)
RL_CS_M_IE              =00000040      164    (2)        #-738    (10)    #-743    (10)
RL_DA                   =00000004      165    (2)        #-727    (10)
RL_DA_M_MRK             =00000001      166    (2)
RL_DA_M_RST             =00000008      168    (2)        #-726    (10)
RL_DA_M_STS             =00000002      167    (2)        #-726    (10)
RL_MP                   =00000006      169    (2)
RL_MP_V_TYP             =00000007      170    (2)        #-749    (10)
RP04$K_LEN               00000032                        #-528    (7)
RP_CS1                  =00000000      173    (2)        #-524    (7)
RP_DT                   =00000018      175    (2)        #-525    (7)
RP_DT_S_DTN             =00000009      177    (2)        #-526    (7)
RP_DT_V_DTN             =00000000      176    (2)        #-526    (7)
SAVABS...               =FFFFFFF0      422    (4)
SCB_BASE                 00000000-XR                       277    (2)
SCB_IMAGE                00000000-XR                       277    (2)
SIZ...                  =00000002      150    (2)        150    (2)
SS$_NORMAL              =00000001                        #-686    (9)
SYS$UNWIND               00000000-XR                       281    (2)
```

T_TYPES	00000007-R	291	(2)	304	(2)	#-540	(7)
UDA_RESPONSE	00000000-XR			283	(2)	969	(10)
UD_INIT	00000000-XR			283	(2)	965	(10)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BTDDEF	1	120 (2)	120 (2)
$DEF	1	150 (2)	
$DEFINI	1	119 (2)	119 (2) 120 (2) 121 (2) 127 (2) 128 (2) 129 (2) 130 (2) 131 (2) 132 (2) 133 (2) 134 (2) 135 (2) 136 (2) 137 (2) 138 (2) 139 (2) 145 (2) 147 (2) 190 (2)
$DS_CHCDEF	1	149 (2)	149 (2)
$DS_CHIDEF	1	150 (2)	150 (2)
$DS_CI_DEF	2	138 (2)	138 (2)
$DS_DISK_DEF	1	139 (2)	139 (2)
$DS_DSADEF	5	145 (2)	145 (2)
$DS_DSDEF	2	146 (2)	146 (2)
$DS_HPODEF	2	127 (2)	127 (2)
$DS_R80_DEF	1	131 (2)	131 (2)
$DS_RA80_DEF	1	137 (2)	137 (2)
$DS_RB730_DEF	1	130 (2)	130 (2)
$DS_RK06_DEF	1	133 (2)	133 (2)
$DS_RK611_DEF	1	129 (2)	129 (2)
$DS_RL01_DEF	1	135 (2)	135 (2)
$DS_RL02_DEF	1	132 (2)	132 (2)
$DS_RL11_DEF	1	134 (2)	134 (2)
$DS_RP04_DEF	1	128 (2)	128 (2)
$DS_UDA50_DEF	1	136 (2)	136 (2)
$DS_WAITUS_S	1	612 (9)	612 (9) 733 (10) 741 (10) 845 (10)
$EQU	1	150 (2)	146 (2) 149 (2)
$EQULS1	1	149 (2)	146 (2) 149 (2)
$EQULST	1	146 (2)	146 (2) 149 (2)
$GBLINI	2		146 (2) 149 (2) 150 (2)
$MSCPDEF	18	121 (2)	121 (2)
$OFFSET	2	418 (4)	418 (4)
$OFFST1	1	422 (4)	422 (4)
$PRDEF	4	119 (2)	119 (2)
$PUSHADR	1	442 (5)	442 (5) 542 (7) 612 (9) 733 (10) 741 (10) 845 (10)
$SSDEF	21	147 (2)	147 (2)
$VIELD	1		150 (2)
$VIELD1	1	150 (2)	150 (2)
CASE	1	440 (5)	440 (5) 599 (8)
CLIDEF	3	118 (2)	118 (2)
ERRSUP_S	1	442 (5)	442 (5) 542 (7)
MODNAM	1	290 (2)	290 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.10	00:00:00.62
Command processing	135	00:00:00.72	00:00:02.25

Pass 1	1165	00:00:20.97	00:00:34.46
Symbol table sort	7	00:00:02.04	00:00:02.51
Pass 2	323	00:00:04.10	00:00:05.46
Symbol table output	46	00:00:00.27	00:00:00.29
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	143	00:0C:01.22	00:00:01.60
Assembler run totals	1859	00:00:29.46	00:00:47.23

The working set limit was 1000 pages.
112476 bytes (220 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1289 non-local and 54 local symbols.
1094 source lines were read in Pass 1, producing 0 object records in Pass 2.
146 pages of virtual memory were used to define 51 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	18
DRB1:[DS.WORK]DS.MLB;218	3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	35

1351 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CONFIG/UPDA=(CONFIG.UPD,CONFIG.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 143 Libraries, Equated Symbols, Macros
(1) 191 Work Psect Declarations ; [13]
(1) 207 Data Psect Declarations ; [13]
(1) 368 VRSetWidth Routine - Set terminal width ; [12]
(1) 446 VRShowWidth Routine - Show terminal width ; [12]
(1) 495 KB_Poll Routine - Keyboard ready bit check
(1) 564 RInput Routine - Flag driven terminal input
(1) 674 RType Routine - Retype input line ('^R' function)
(1) 715 Out Routines - Flag driven terminal output
(1) 833 WriteVBlk Routine - Console write driver
(1) 880 ReadVBlk Routine - Console read driver
(1) 1075 SIZE_TERM Routine to size the console terminal

```
0000     1          .TITLE  CONSOLE *** CONSOLE Console I/O handler
0000     2          .IDENT  /07-23/
0000     3          .NoShow Conditionals    ;                                                    [16]
0000     4
0000     5 ;++
0000     6 ; COPYRIGHT (C) 1977, 1983
0000     7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     8 ;
0000     9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    15 ; REMAIN IN DEC.
0000    16 ;
0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    19 ; CORPORATION.
0000    20 ;
0000    21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    23 ;--
```

```
                        25 ;++
                        26 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
                        27 ;
                        28 ; ABSTRACT:
                        29 ;
                        30 ;       This routine contains the interface with the LSI-11 console.
                        31 ;       It provides the routines necessary to read and write characters
                        32 ;       and strings on the operators console.  It contains the limited
                        33 ;       functionality to interface with QIO.
                        34 ;
                        35 ; ENVIRONMENT:
                        36 ;
                        37 ;       Operates as part of the Supervisor only in stand-alone.
                        38 ;       Portions hereof must operate in KERNEL mode.
                        39 ;
                        40 ; AUTHOR:
                        41 ;
                        42 ;       KEN CHAPMAN     10-NOV-77       VERSION 01.
                        43 ;
                        44 ; MODIFICATIONS:
                        45 ;
                        46 ;       01      NICK HOWGATE    30-DEC-77       VERSION 02.
                        47 ;               CMK$TCONSOLE - TURN OFF ALL OUTPUT IF APT REQUESTS
                        48 ;               NO REPORT (DSA$GL_FLAGS)=DSA$V_NORPT
                        49 ;
                        50 ;       02      Roger Riggs     25-JUL-78
                        51 ;               CMK$TCONSOLE looks also at DS$V_CTRLO in DS$GL_FLAGS
                        52 ;               MAKE ^O WORK.
                        53 ;       03      Made unsolicited input is echoed as bells instead of the
                        54 ;               characters typed.
                        55 ;
                        56 ;       04      NICK HOWGATE    5-FEB-79        (ESSAA-5.00)
                        57 ;               CONVERT TABS TO SPACES
                        58 ;
                        59 ;       05      Implement P4 buffer in WRITEBLK
                        60 ;
                        61 ;       06      Fix IO$_READPROMPT to always prompt on new line
                        62 ;
                        63 ;       07      Defer echo of ^C until KB_CHECK but suppress rest of output
                        64 ;               until ^C is serviced
                        65 ;
                        66 ;       08      MARION BAGGETT  NOV 12,1979
                        67 ;               ADD THE CONTROL S AND CONTROL Q FUNCTION FOR STARTING AND
                        68 ;               STOPPING THE PRINTOUT ON THE TERMINAL.
                        69 ;
                        70 ;       09      Roger Riggs, 14-Jan-1980, Version 5.2
                        71 ;               Added checking of IO$M_CANCTRLO  on write to terminal.
                        72 ;               Used by exception to make sure exceptions always print.
                        73 ;               Also fixed RINPUT to ignore character if bit 15 (ERR) is
                        74 ;               Set in the data register.
                        75 ;
                        76 ;       10      Dave Butenhof, 17-oct-1980, version 6.1
                        77 ;               Cause re-type of input line ('^R') after asynchronous output
                        78 ;               occurs, in the fashion of VMS.  Also, remove hyphen line
                        79 ;               continuation code, since it doesn't belong here.
                        80 ;
                        81 ;       11      To fully support ^C 'disable' in tight time-critical code,
```

```
                                0000     82 ;               cut down on instruction stream for control characters.
                                0000     83 ;               Specifically, test for ^C first in RINPUT instead of last.
                                0000     84 ;               Also, make new routine OUT_CNTRL to output ^C and ^O echo;
                                0000     85 ;               without all the overhead of doing extra KB_POLL each
                                0000     86 ;               character (small messages), or checking for tabs, etc.
                                0000     87 ;
                                0000     88 ;       12      Dave Butenhof, 17-mar-1981, version 6.3
                                0000     89 ;               Add VRSETWIDTH routine, and hooks for SET WIDTH command.
                                0000     90 ;               This allows users (FS in particular) to avoid destroying
                                0000     91 ;               80 column printer paper with output from DS. Also add
                                0000     92 ;               VRSHOWWIDTH routine.
                                0000     93 ;
                                0000     94 ;       13      - Jack Stansbury, 22-Oct-1981, Version 6.-
                                0000     95 ;               Changed SEP Psect to CODE, DATA, and WORK.
                                0000     96 ;               Also added .LIBRARY statements for $DS and $DIAG.
                                0000     97 ;
                                0000     98 ;       14      - Jack Stansbury, 26-Oct-1981, Version 6.-
                                0000     99 ;               Fixed more truncation errors....
                                0000    100 ;
                                0000    101 ;       15      - Jack Stansbury, 25-Jan-1982, Version 6.6
                                0000    102 ;               Changed the WIDTH variable to make it global, so that QA can
                                0000    103 ;               access it.
                                0000    104 ;
                                0000    105 ;       16      - Jack Stansbury, 23-Mar-1982, Version 6.?
                                0000    106 ;               Added typecoding to the output messages.
                                0000    107 ;
                                0000    108 ;       17      - Richard Brown, 8-June-82, Version 6.8
                                0000    109 ;               Addition of support for specification of
                                0000    110 ;               terminators other than carriage-return.  We will
                                0000    111 ;               now be functionally equivalent to VMS QIO calls
                                0000    112 ;               with respect to the optional terminator specification
                                0000    113 ;               (argument P4) for the terminal driver.
                                0000    114 ;
                                0000    115 ;       18      Bob Bergazzi    6-Aug-82        Version 6.8
                                0000    116 ;               Added code to change a ^Z into an EXIT command
                                0000    117 ;
                                0000    118 ;       19      Jack Stansbury, 26-Oct-1982, Version 6.9
                                0000    119 ;               Added code to echo the ^C character only if the
                                0000    120 ;               DS$V_Ctrl_C_No_Echo bit is clear. If it is set,
                                0000    121 ;               then ^C's will not be echoed.
                                0000    122 ;
                                0000    123 ;       20      Bob Bergazzi    5-March-1983    Version 6.11
                                0000    124 ;               Added code to make SET WIDTH work online.
                                0000    125 ;
                                0000    126 ;       21      Bob Bergazzi    8-Apr-1983      Version 6.11
                                0000    127 ;               Removed incorrectly implemented edit 18.
                                0000    128 ;
                                0000    129 ;       22      Domenic Andella 25-Apr-83       Version 6.11
                                0000    130 ;               Added code in CMK$TCONSOLE routine to activate
                                0000    131 ;               11/XXX console terminal line.
                                0000    132 ;
                                0000    133 ;       23      Richard Brown   22-June-83      Version 6.12
                                0000    134 ;               Added routine SIZE_TERM, which determines
                                0000    135 ;               console terminal type.
                                0000    136 ;
                                0000    137 ;               Also removed code in READVBLK that treated a
                                0000    138 ;               question mark as a terminator, since many
```

```
                                0000   139 ;                   terminal IDs contain question marks, and we
                                0000   140 ;                   couldn't figure why it was a terminator anyway.
                                0000   141 ;--
```

```
                        0000   143          .SBTTL  Libraries, Equated Symbols, Macros
                        0000   144 ;
                        0000   145 ; INCLUDE FILES:
                        0000   146 ;
                        0000   147
                        0000   148          .Library        /Sys$Library:Lib/        ;                       [16]
                        0000   149          .LIBRARY        /$DS/                   ;                       [13]
                        0000   150          .LIBRARY        /$DIAG/                 ;                       [13]
                        0000   151
                        0000   152 ;
                        0000   153 ; MACROS:
                        0000   154 ;
                        0000   155
                        0000   156          $DS_DSADEF
                        C000   157          CLIDEF                          ; CLI data structure            [12]
                        0000   158          $PRDEF
                        0000   159          $QIODEF
                        0000   160          CMKDEF
                        0000   161          DSFDEF
                        0000   162          $DS_TypeDef                     ;                               [16]
                        0000   163          CRDDef                          ; CRD Bit definitions           [19]
                        0000   164          $TTDEF                          ; Terminal characteristics      [23]
                        0000   165
                        0000   166 ;
                        0000   167 ; EQUATED SYMBOLS:
                        0000   168 ;
                        0000   169
00000080                0000   170          PARITY  = 1@7
00000020                0000   171          LOWCASE = 1@5
00000003                0000   172          CNTR'.C = 3
00000007                0000   173          BELL    = 7
00000008                0000   174          BS      = 8                     ;                               [17]
00000009                0000   175          TAB     = 9
0000000A                0000   176          LF      = 10
0000000B                0000   177          VT      = 11                    ;                               [17]
0000000C                0000   178          FF      = 12                    ;                               [17]
0000000D                0000   179          CR      = 13
0000000F                0000   180          CNTRLO  = 15
00000011                0000   181          CNTRLQ  = 17
00000012                0000   182          CNTRLR  = 18
00000013                0000   183          CNTRLS  = 19
00000015                0000   184          CNTRLU  = 21
00000020                0000   185          SPACE   = 32
0000002D                0000   186          HYPHEN  = 45
0000003F                0000   187          QUES'N  = 63
0000005C                0000   188          BSLASH  = 92
0000007F                0000   189          DELETE  = 127
```

```
                                     0000   191          .SUBTITLE       Work Psect Declarations         ;                                [13]
                                00000000    192          .PSECT  Work, NoExe, NoShr, Wrt, Long        ;                                [13]
                                     0000   193 ;
                                     0000   194 ; OWN STORAGE:
                                     0000   195 ;
                                     0000   196
                                  00 0000   197 COLUMN:          .BYTE   0                       ; COLUMN COUNTER
                                     0001   198
                                  00 0001   199 LASTCHAR:        .BYTE   0                       ; Last character output to console
                                     0002   200
                                  50 0002   201 DS$GB_WIDTH::    .BYTE   80                      ; Initial width is 80            [15]
                                     0003   202
                                  00 0003   203 CONSFLG:         .BYTE   0                       ; CONTROL S AND CONTROL Q FLAG
                                     0004   204
                            00000000 0004   205 EVENT_FLAGS:     .LONG   0                       ; Receives event flags           [23]
```

```
                                    0008   207          .SUBTITLE       Data Psect Declarations         ;                       [13]
                                00000000   208          .PSECT  Data, NoExe, Shr, NcWrt, Long        ;                       [13]
                                    0000   209
                                    0000   210 T_CNTRLC:
                  0A 0D 43 5E 0A 0D 00'     0000   211          .ASCIC  <CR><LF>'^C'<CR><LF>
                                    06      0000
                                    0007   212
                                    0C 7   213 T_CNTRLU:
                     0A 0D 55 5E 00'  0C 07   214          .ASCIC  '^U'<CR><LF>
                                    04      0C 07
                                    000C   215
                                    000C   216 T_CNTRLR:
                        52 5E 00'   000C   217          .ASCIC  '^R'
                                    02      000C
                                    C00F   218
                                    000F   219 T_CNTRLO:
                     0A 0D 4F 5E 00'  000F   220          .ASCIC  '^O'<CR><LF>
                                    04      000F
                                    0014   221
                                    0014   222 T_CRLF:
                        0A 0D 00'   0014   223          .ASCIC  <CR><LF>
                                    02      0014
                                    0017   224
                                    0017   225 T_WIDTH:                                                ;       [12]
20 74 73 75 6D 20 68 74 64 69 57 00' 0017   226          .ASCIC  "Width must be greater than 0 and less than 133!/"      ;       [12]
74 20 72 65 74 61 65 72 67 20 65 62  0023
65 6C 20 64 6E 61 20 30 20 6E 61 68  002F
21 33 33 31 20 6E 61 68 74 20 73 73  003B
                                    2F  0047
                                    30  0017
                                    0048   227
                                    0048   228 T_SHOW_WIDTH:                                            ;       [12]
69 77 20 6C 61 6E 69 6D 72 65 54 00' 0048   229          .ASCIC  "Terminal width is !ZB character!%S!/"      ;       [12]
63 20 42 5A 21 20 73 69 20 68 74 64  0054
21 53 25 21 72 65 74 63 61 72 61 68  0060
                                    2F  006C
                                    24  0048
                                    006D   230
                                    006D   231 ;                                                                [23]
                                    006D   232 ;TABLE OF VIDEO TERMINAL TYPES                                   [23]
                                    006D   233 ;                                                                [23]
                                    006D   234
                                    006D   235 VIDEO_TYPES:
                                    006D   236
                                    006D   237 T_VT50_A:
                     41 2F 1B 00'   006D   238          .ASCIC  <27>"/A"
                                    03      006D
                                    40      0071   239          .BYTE   TT$_VT5X
                                    0072   240 T_VT50_B:
                     42 2F 1B 00'   0072   241          .ASCIC  <27>"/B"
                                    03      0072
                                    40      0076   242          .BYTE   TT$_VT5X
                                    0077   243 T_VT50_C:
                     43 2F 1B 00'   0077   244          .ASCIC  <27>"/C"
                                    03      0077
                                    40      007B   245          .BYTE   TT$_VT5X
                                    007C   246 T_VT55_E:
```

```
                45 2F 1B 00'  007C    247          .ASCIC  <27>"/E"
                      03      007C
                      41      0080    248          .BYTE   TT$_VT55
                              0081    249 T_VT50_H:
                48 2F 1B 00'  0081    250          .ASCIC  <27>"/H"
                      03      0081
                      40      0085    251          .BYTE   TT$_VT5X
                              0086    252 T_VT50_J:
                4A 2F 1B 00'  0086    253          .ASCIC  <27>"/J"
                      03      0086
                      40      008A    254          .BYTE   TT$_VT5X
                              008B    255 T_VT52_K:
                4B 2F 1B 00'  008B    256          .ASCIC  <27>"/K"
                      03      008B
                      40      C08F    257          .BYTE   TT$_VT52
                              0090    258 T_VT52_L:
                4C 2F 1B 00'  0090    259          .ASCIC  <27>"/L"
                      03      0090
                      40      0094    260          .BYTE   TT$_VT52
                              0095    261 T_VT52_M:
                4D 2F 1B 00'  0095    262          .ASCIC  <27>"/M"
                      03      0095
                      40      0099    263          .BYTE   TT$_VT52
                              009A    264 T_VT100A:
                5A 2F 1B 00'  009A    265          .ASCIC  <27>"/Z"
                      03      009A
                      60      009E    266          .BYTE   TT$_VT100
                              009F    267 ;T_VT61:
                              009F    268 ;        .ASCIC  <27>@/'@
                              009F    269 ;        .BYTE   TT$_VT61
                              009F    270 ;T_VT71:
                              009F    271 ;        .ASCIC  <27><47><126>
                              009F    272 ;        .BYTE   TT$_VT71
                              009F    273 T_VT100:
63 30 3B 31 3F 5B 1B 00'      009F    274          .ASCIC  <27>"[?1;0"<99>
                      07      009F
                      60      00A7    275          .BYTE   TT$_VT100
                              00A8    276 T_VT100B:
63 31 3B 31 3F 5B 1B 00'      00A8    277          .ASCIC  <27>"[?1;1"<99>
                      07      00A8
                      60      00B0    278          .BYTE   TT$_VT100
                              00B1    279 T_VT100C:
63 32 3B 31 3F 5B 1B 00'      00B1    280          .ASCIC  <27>"[?1;2"<99>
                      07      00B1
                      60      00B9    281          .BYTE   TT$_VT100
                              00BA    282 T_VT100D:
63 33 3B 31 3F 5B 1B 00'      00BA    283          .ASCIC  <27>"[?1;3"<99>
                      07      00BA
                      60      00C2    284          .BYTE   TT$_VT100
                              00C3    285 T_VT100E:
63 34 3B 31 3F 5B 1B 00'      00C3    286          .ASCIC  <27>"[?1;4"<99>
                      07      00C3
                      60      00CB    287          .BYTE   TT$_VT100
                              00CC    288 T_VT100F:
63 35 3B 31 3F 5B 1B 00'      00CC    289          .ASCIC  <27>"[?1;5"<99>
                      07      00CC
                      60      00D4    290          .BYTE   TT$_VT100
```

```
                                   00D5   291 T_VT100G:
                63 36 3B 31 3F 5B 1B 00' 00D5   292         .ASCIC  <27>"[?1;6"<99>
                                      07   00D5
                                      60   00DD   293         .BYTE   TT$_VT100
                                           00DE   294 T_VT100H:
                63 37 3B 31 3F 5B 1B 00' 00DE   295         .ASCIC  <27>"[?1;7"<99>
                                      07   00DE
                                      60   00E6   296         .BYTE   TT$_VT100
                                           00E7   297 T_VT220:
37 3B 32 3B 31 3B 32 36 3F 5B 1B 00' 00E7   298         .ASCIC  <27>"[?62;1;2;7;8"<99>
                           63 38 3B  00F3
                                      0E   00E7
                                      00   00F6   299         .BYTE   TT$_UNKNOWN
                                           00F7   300 T_VT240:
33 3B 32 3B 31 3B 32 35 3F 5B 1B 00' 00F7   301         .ASCIC  <27>"[?52;1;2;3;4;7;8"<99>
                  63 38 3B 37 3B 34  38   0103
                                      12   00F7
                                      00   010A   302         .BYTE   TT$_UNKNOWN
                                           010B   303 ;T_VT132:
                                           010B   304 ;        .ASCIC  <27>"[?4;0"<99>
                                           010B   305 ;        .BYTE   TT$_VT132
                                           010B   306 ;T_VT132A:
                                           010B   307 ;        .ASCIC  <27>"[?4;1"<99>
                                           010B   308 ;        .BYTE   TT$_VT132
                                           010B   309 ;T_VT132B:
                                           010B   310 ;        .ASCIC  <27>"[?4;2"<99>
                                           010B   311 ;        .BYTE   TT$_VT132
                                           010B   312 ;T_VT132C:
                                           010B   313 ;        .ASCIC  <27>"[?4;3"<99>
                                           010B   314 ;        .BYTE   TT$_VT132
                                           010B   315 ;T_VT132D:
                                           010B   316 ;        .ASCIC  <27>"[?4;4"<99>
                                           010B   317 ;        .BYTE   TT$_VT132
                                           010B   318 ;T_VT132E:
                                           010B   319 ;        .ASCIC  <27>"[?4;5"<99>
                                           010B   320 ;        .BYTE   TT$_VT132
                                           010B   321 ;T_VT132F:
                                           010B   322 ;        .ASCIC  <27>"[?4;6"<99>
                                           010B   323 ;        .BYTE   TT$_VT132
                                           010B   324 ;T_VT132G:
                                           010B   325 ;        .ASCIC  <27>"[?4;7"<99>
                                           010B   326 ;        .BYTE   TT$_VT132
                                           010B   327 ;T_VT61A:
                                           010B   328 ;        .ASCIC  <27><47><97>
                                           010B   329 ;        .BYTE   TT$_VT61
                                           010B   330 ;T_VT61B:
                                           010B   331 ;        .ASCIC  <27><47><98>
                                           010B   332 ;        .BYTE   TT$_VT61
                                           010B   333 ;T_VT61C:
                                           010B   334 ;        .ASCIC  <27><47><99>
                                           010B   335 ;        .BYTE   TT$_VT61
                                           010B   336 END_VIDEO_TYPES:
                                FFFFFFFF   010B   337         .LONG -1
                                           010F   338
                                           010F   339 ;                                                          [23]
                                           010F   340 ; TABLE OF HARDCOPY TERMINAL TYPES                         [23]
                                           010F   341 ;                                                          [23]
```

```
                                010F    342
                                010F    343 HARDCOPY_TYPES:
                                010F    344
                                010F    345 T_LA12:
      63 35 31 3F 5B 1B 00'     010F    346             .ASCIC  <27>"[?15"<99>
                                06      010F
                                24      0116    347             .BYTE   TT$_LA12
                                0117    348 T_LA100:
      63 30 31 3F 5B 1B 00'     0117    349             .ASCIC  <27>"[?10"<99>
                                06      0117
                                25      011E    350             .BYTE   TT$_LA100
                                011F    351 T_LA120:
         63 32 3F 5B 1B 00'     011F    352             .ASCIC  <27>"[?2"<99>
                                05      011F
                                21      C125    353             .BYTE   TT$_LA120
                                0126    354 T_LA34:
   63 30 3B 33 3F 5B 1B 00'     0126    355             .ASCIC  <27>"[?3;0"<99>
                                07      0126
                                22      012E    356             .BYTE   TT$_LA34
                                012F    357 T_LA34A:
   63 31 3B 33 3F 5B 1B 00'     012F    358             .ASCIC  <27>"[?3;1"<99>
                                07      012F
                                22      0137    359             .BYTE   TT$_LA34
                                0138    360 END_HARDCOPY_TYPES:
                   FFFFFFFF     0138    361             .LONG -1
                                013C    362
                                013C    363 T_LA36:                             ; LA36 has no response sequence.        [23]
                                20      013C    364             .BYTE   TT$_LA36
                                013D    365 ;
                                013D    366
```

```
                     013D   368          .SBTTL  VRSetWidth Routine - Set terminal width                 ;       [12]
                00000000    369          .PSECT  Code, Exe, Shr, NoWrt, Long                            ;       [13]
                    0000    370 ;++                                                                             [12]
                    0000    371 ; VRSETWIDTH                                                                    [12]
                    0000    372 ;                                                                               [12]
                    0000    373 ; Functional description:                                                       [12]
                    0000    374 ;                                                                               [12]
                    0000    375 ;       This is a CLI command routine.  It loads the global DS$GB_WIDTH with    [15]
                    0000    376 ;       a value specified to CLI.  If the width is out of range, an error       [12]
                    0000    377 ;       message is typed and it returns a failure status code (though CLI       [12]
                    0000    378 ;       does not currently recognize return status).                            [12]
                    0000    379 ;                                                                               [12]
                    0000    380 ;       In online mode, a QIO SENSE mode operation is done in order to get      [20]
                    0000    381 ;       the terminal's characteristics, and then a SET is done with the width   [20]
                    0000    382 ;       parameter changed.                                                      [20]
                    0000    383 ;                                                                               [12]
                    0000    384 ; Calling sequence:                                                             [12]
                    0000    385 ;                                                                               [12]
                    0000    386 ;       JSB     VRSETWIDTH                                                      [12]
                    0000    387 ;                                                                               [12]
                    0000    388 ; Input Parameters:                                                             [12]
                    0000    389 ;                                                                               [12]
                    0000    390 ;       none                                                                    [12]
                    0000    391 ;                                                                               [12]
                    0000    392 ; Implicit Inputs:                                                              [12]
                    0000    393 ;                                                                               [12]
                    0000    394 ;       R2      points to CLI data table                                        [12]
                    0000    395 ;                                                                               [12]
                    0000    396 ; Output parameters:                                                            [12]
                    0000    397 ;                                                                               [12]
                    0000    398 ;       none                                                                    [12]
                    0000    399 ;                                                                               [12]
                    0000    400 ; Implicit outputs:                                                             [12]
                    0000    401 ;                                                                               [12]
                    0000    402 ;       none                                                                    [12]
                    0000    403 ;                                                                               [12]
                    0000    404 ; Side effects:                                                                 [12]
                    0000    405 ;                                                                               [12]
                    0000    406 ;       Changes value of DS$GB_WIDTH variable                                   [15]
                    0000    407 ;                                                                               [12]
                    0000    408 ; Completion codes:                                                             [12]
                    0000    409 ;                                                                               [12]
                    0000    410 ;       0       failure (value out of range)                                    [12]
                    0000    411 ;       1       success                                                         [12]
                    0000    412 ;--                                                                             [12]
```

```
                                0000  VRSETWIDTH::                                        ;                                       [12]
              50    1C A2   D0  0000            MOVL    CLI$L_DATA(R2), R0                ; Get width                             [12]
                       70   15  0004            BLEQ    10$                               ; Branch if value too small             [12]
       00000084 8F     50   D1  0006            CMPL    R0, #132                          ; Also branch if                        [12]
                       67   14  000D            BGTR    10$                               ;  value too large                      [12]
       00000002'EF     50   90  000F            MOVB    R0, DS$GB_WIDTH                   ; Store the value                       [15]
                 50    01   D0  0016            MOVL    #1, R0                            ; Return success                        [12]
                                0019            Br_If_Not_User  5$                        ; Thats all for standalone              [20]
                                0021
                       7E   7C  0021            ClrQ    -(SP)                             ; Space for characteristics buffer      [20]
                 50    6E   7E  0023            MovAQ   (SP), R0                          ; on stack                              [20]
                                0026            $QIOW_S , -                               ; First, read the terminal characteristics
                                0026                    L^DS$GW_TTOUT, -                  ; Channel number                        [20]
                                0026                    #IO$_SENSEMODE, -                 ; Function                              [20]
                                0026                    P1 = (R0)                         ; Address of characteristics buffer     [20]
                                0047
  02 AE    00000002'EF          9B  0047            MovZBW  DS$GB_WIDTH, 2(SP)                ; Set the desired page width            [20]
                 50    6E   7E  004F            MovAQ   (SP), R0                          ; Buffer on the stack                   [20]
                                0052            $QIOW_S , -                               ; no efn                                [20]
                                0052                    L^DS$GW_TTOUT, -                  ; Channel number                        [20]
                                0052                    #IO$_SETMODE, -                   ; Function                              [20]
                                0052                    P1 = (R0)                         ; Address of characteristics buffer     [20]
                       8E   7C  0073            ClrQ    (SP)+                             ; Release the buffer on the stack       [20]
                            05  0075  5$:       RSB                                       ;                                       [12]
                                0076
                                0076  10$:      $Print  -                                 ; Print with a typecode                 [16]
                                0076                      #DS$K_Type_Command_Err, -       ; ... the typecode                      [16]
                                0076                      #DS$K_PrintF, -                 ; ... use PrintF to print it            [16]
                                0076                      T_Width                         ; ... the message                       [16]
                       50   D4  0089            CLRL    R0                                ; Return failure                        [12]
                            05  008B            RSB                                       ;                                       [12]
```

```
008C    446          .SBTTL  VRShowWidth Routine - Show terminal width      ;                [12]
008C    447
008C    448 ;++                                                                             [12]
008C    449 ; VRSHOWWIDTH                                                                   [12]
008C    450 ;                                                                               [12]
008C    451 ; Functional description:                                                       [12]
008C    452 ;                                                                               [12]
008C    453 ;       This is a CLI command routine.  It prints the current DS$GB_WIDTH.      [12]
008C    454 ;                                                                               [12]
008C    455 ; Calling sequence:                                                             [12]
008C    456 ;                                                                               [12]
008C    457 ;       JSB     VRSHOWWIDTH                                                     [12]
008C    458 ;                                                                               [12]
008C    459 ; Input Parameters:                                                             [12]
008C    460 ;                                                                               [12]
008C    461 ;       none                                                                    [12]
008C    462 ;                                                                               [12]
008C    463 ; Implicit Inputs:                                                              [12]
008C    464 ;                                                                               [12]
008C    465 ;       none                                                                    [12]
008C    466 ;                                                                               [12]
008C    467 ; Output parameters:                                                            [12]
008C    468 ;                                                                               [12]
008C    469 ;       none                                                                    [12]
008C    470 ;                                                                               [12]
008C    471 ; Implicit outputs:                                                             [12]
008C    472 ;                                                                               [12]
008C    473 ;       none                                                                    [12]
008C    474 ;                                                                               [12]
008C    475 ; Side effects:                                                                 [12]
008C    476 ;                                                                               [12]
008C    477 ;       none                                                                    [12]
008C    478 ;                                                                               [12]
008C    479 ; Completion codes:                                                             [12]
008C    480 ;                                                                               [12]
008C    481 ;       1       success                                                         [12]
008C    482 ;--                                                                             [12]
```

```
                     008C    484 VRSHOWWIDTH::                                       ;                                                   [12]
          50 00000002'EF 9A  008C    485         MovZBL  L^DS$GB_Width, R0           ; Put the current width in R0                       [16]
                     0093    486         $Print  -                                   ; Print with a typecode                             [16]
                     0093    487                   #DS$K_Type_Command_Out, -         ;     ; ... the typecode number                     [16]
                     0093    488                   #DS$K_PrintF, -                   ; ... the type of print routine                     [16]
                     0093    489                   T_Show_Width, -                   ; ... the format message                            [16]
                     0093    490                   R0                                ; ... and the width                                 [16]
                     00A8    491
             50 01   D0  00A8    492         MOVL    #1, R0                          ; Set success                                       [12]
                 05  00AB    493         RSB                                         ;                                                   [12]
```

```
                     00AC    495          .SBTTL  KB_Poll Routine - Keyboard ready bit check
                     00AC    496 ;++
                     00AC    497 ; FUNCTIONAL DESCRIPTION:
                     00AC    498 ;
                     00AC    499 ;       THIS ROUTINE CHECKS THE 'READY' BIT IN THE CONSOLE'S STATUS
                     00AC    500 ;       REGISTER AND, IF CLEAR, RETURNS TO THE CALLING ROUTINE.
                     00AC    501 ;       If no character is ready this returns immediately.
                     00AC    502 ;       Select one of the following based on the character typed:
                     00AC    503 ;       ^C      clear local flags and set the global ^C flag
                     00AC    504 ;               set ^O flag to suppress output until ^C is serviced by KB_CHECK
                     00AC    505 ;       ^O      toggle the ^O bit (output may be flushed based on this bit)
                     00AC    506 ;       ^S      SET CONSFLG FLAG TO SUSPEND TERMINAL OUTPUT
                     00AC    507 ;       ^Q      CLEAR CONSFLG FLAG TO RESUME TERMINAL OUTPUT IF ANY
                     00AC    508 ;       ^C      WILL ALSO CLEAR CONSFLG FLAG.
                     00AC    509 ;       Else,   echo the character as a <BELL>
                     00AC    510 ;
                     00AC    511 ; CALLING SEQUENCE:
                     00AC    512 ;
                     00AC    513 ;       SUBROUTINE CALL.
                     00AC    514 ;
                     00AC    515 ; INPUT PARAMETERS:
                     00AC    516 ;
                     00AC    517 ;       NONE
                     00AC    518 ;
                     00AC    519 ; IMPLICIT INPUTS:
                     00AC    520 ;
                     00AC    521 ; OUTPUT PARAMETERS:
                     00AC    522 ;
                     00AC    523 ;       DS$GL_FLAGS<DS$V_CTRLC>         = ^C FLAG.
                     00AC    524 ;
                     00AC    525 ; IMPLICIT OUTPUTS:
                     00AC    526 ;
                     00AC    527 ;       NONE
                     00AC    528 ;
                     00AC    529 ; COMPLETION CODES:
                     00AC    530 ;
                     00AC    531 ;       NONE
                     00AC    532 ;
                     00AC    533 ; SIDE EFFECTS:
                     00AC    534 ;
                     00AC    535 ;       NONE
                     00AC    536 ;--
```

```
                        00AC    538 KB_POLL::
                03   BB 00AC    539         PUSHR   #^M<R0,R1>                ; SAVE R0 & R1.
                           00AE    540         CMK     RCONSOLE
                           00BD    541
          1C 50 07   E1 00BD    542         BBC     #7,R0,KB_POLL_X         ; EXIT IF NO CHARACTER
                           00C1    543
             51 11   91 00C1    544         CMPB    #CNTRLQ,R1
                17   13 00C4    545         BEQL    KB_POLL_X               ; SKIP ECHO IF CONTROL Q
                0F   19 00C6    546         BLSS    10$                     ; SKIP FOR SPEED OPTIMIZATION
                           00C8    547
             51 03   91 00C8    548         CMPB    #CNTRLC,R1
                10   13 00CB    549         BEQL    KB_POLL_X               ; SKIP ECHO IF CONTROL-C
                           00CD    550
             51 0F   91 00CD    551         CMPB    #CNTRLO,R1
                0B   13 00D0    552         BEQL    KB_POLL_X               ; SKIP ECHO IF CONTROL-O
                           00D2    553
             51 13   91 00D2    554         CMPB    #CNTRLS,R1
                06   13 00D5    555         BEQL    KB_POLL_X               ; SKIP ECHO IF CONTROL S
                           00D7    556
             51 07   D0 00D7    557 10$:    MOVL    #BELL,R1                ; ECHO A BELL.
              015D   30 00DA    558         BSBW    OUT_ECHO                ; TYPE IT OUT
                           00DD    559
                           00DD    560 KB_POLL_X:
                03   BA 00DD    561         POPR    #^M<R0,R1>              ; RESTORE R0 & R1.
                05   00DF    562         RSB                             ; RETURN.
```

```
                        564          .SBTTL  RInput Routine - Flag driven terminal input
                        565 ;++
                        566 ; FUNCTIONAL DESCRIPTION:
                        567 ;
                        568 ;       This routine returns a character from the console terminal.
                        569 ;       The associated interrupt routine CMK$RCONSOLE checks the
                        570 ;       the character and sets or cleas flags for ^S, ^Q, ^O.
                        571 ;
                        572 ; CALLING SEQUENCE:
                        573 ;
                        574 ;       BSBW    RINPUT
                        575 ;
                        576 ; INPUT PARAMETERS:
                        577 ;
                        578 ;       NONE
                        579 ;
                        580 ; IMPLICIT INPUTS:
                        581 ;
                        582 ;       NONE
                        583 ;
                        584 ; OUTPUT PARAMETERS:
                        585 ;
                        586 ;       NONE
                        587 ;
                        588 ; IMPLICIT OUTPUTS:
                        589 ;
                        590 ;       NONE
                        591 ;
                        592 ; COMPLETION CODES:
                        593 ;
                        594 ;       NONE
                        595 ;
                        596 ; SIDE EFFECTS:
                        597 ;
                        598 ;       NONE
                        599 ;--
```

```
                                00E0   601 RINPUT:                                        ; KEYBOARD INPUT ROUTINE.
00000000'EF   17   E5   00E0   602          BBCC    S^#DS$V_OUTPUT, -
                03     00E7   603                   L^DS$GL_FLAGS, 10$     ; Branch if no asynch output, clear flag
              00D3   30   00E8   604          BSBW    RTYPE                  ; Re-type input line
                          00EB   605
                          00EB   606 10$:     CMK     RCONSOLE               ; READ KEYBOARD.
     E2 50    07   E1   00FA   607          BBC     #7,R0,RINPUT           ; TRY AGAIN IF NOTHING THERE
00   51  04    08   ED   00FE   608          CMPZV   #8,#4,R1,#0            ; CONSOLE DATA?
            DB   12   0103   609          BNEQ    RINPUT                 ; NO, IGNORE IT
         50   51   D0   0105   610          MOVL    R1,R0                  ; COPY CHARACTER
                          0108   611
                          0108   612 RINPUTX:                                       ; ROUTINE EXIT POINT.
                05   0108   613          RSB                            ; RETURN TO CALLING ROUTINE
                          0109   614
                          C109   615 CMK$RCONSOLE::
                51   D4   0109   616 5$:      CLRL    R1                     ; INITIALIZE R1.                        [19]
                          010B   617
         50     20   DB   010B   618 10$:     MFPR    #PR$_RXCS, R0          ; GET KEYBOARD STATUS.
   03    50     07   E0   010E   619          BBS     #7, R0, 15$            ;
              00A8   31   0112   620          BRW     90$                    ; No character, exit'
                          0115   621
         51     21   DB   0115   622 15$:     MFPR    #PR$_RXDB, R1          ; INPUT CHARACTER.
      EF 51     0F   F0   0118   623          BBS     #15,R1,10$             ; Error on character, try again.
                          011C   624
   51   0F00   8F   B3   011C   625 20$:     BITW    #^XF00,R1              ; Is this a console character?
                03   13   0121   626          BEQL    21$                    ; Yes
              0097   31   0123   627          BRW     90$                    ; No - just REI
                          0126   628
51 51   07     00   EF   0126   629 21$:     EXTZV   #0,#7,R1,R1            ; Remove parity and garbage
           51   03   91   012B   630          CMPB    #CNTRLC,R1             ; CONTROL-C?
                48   12   012E   631          BNEQ    25$                    ; Check next
      00000003'EF   94   0130   632          CLRB    CONSFLG                ; RESUME PRINTING ON ^C
                          0136   633
                          0136   634          Br_If_Ctrl_C_No_Echo    22$    ; If not echoing ^C's, skip around     [19]
                03   BB   013E   635          PUSHR   #^M<R0,R1>             ; Save flag and character
50   00000000'EF   9E   0140   636          MOVAB   T_CNTRLC,R0            ; ASCIC <NL>^C<NL>
              0CAC   30   0147   637          BSBW    OUT_CNTRL              ; Type it
                03   BA   014A   638          POPR    #^M<R0,R1>             ; Restore
                          014C   639          Set_CtrlO                      ; Set ^O flag                           [16]
                          0154   640          Set_CtrlC                      ; Set ^C flag                           [16]
                5F   11   015C   641          BRB     90$                    ; Return Control C character
                          015E   642
                          015E   643 22$:     ; We are not echoing ^C's. Now check to see if we are allowing ^C as a  [19]
                          015E   644          ; valid character. If we are not, then go back up and see if there is   [19]
                          015E   645          ; another character there.                                              [19]
                          015E   646
                          015E   647          Br_If_Flush_Ctrl_C      5$     ; If flushing, return with no character [19]
                          0166   648          Set_Flush_Ctrl_C               ; Not currently flushing -> start now,  [19]
                          016E   649          Set_CtrlC                      ; . . . set ^C flag, and                [19]
                45   11   0176   650          BRB     90$                    ; . . . return with the ^C character    [19]
                          0178   651
                          0178   652 25$:     Clear_Flush_Ctrl_C             ; This is not a ^C - stop flushing now  [19]
         51     13   91   0180   653          CMPB    #CNTRLS,R1             ; CONTROL-S ?
                09   12   0183   654          BNEQ    28$                    ; NO, NEXT
00000003'EF     01   88   0185   655          BISB    #1,CONSFLG             ; STOP PRINTING
                2D   11   018C   656          BRB     80$                    ; Flush and exit
                          018E   657
```

```
                51   11   91   018E   658 28$:    CMPB    #CNTRLQ,R1                  ; CONTROL-Q ?
                     08   12   0191   659         BNEQ    30$                         ; NO EXIT
          00000003'EF   94   0193   660         CLRB    CONSFLG                     ; RESUME PRINTING
                     20   11   0199   661         BRB     80$                         ; Flush and exit
                                0198   662
                51   0F   B1   019B   663 30$:    CMPW    #CNTRLO,R1                  ; CONTROL O?
                     1D   12   019E   664         BNEQ    90$                         ; NO
          00000003'EF   94   01A0   665         CLRB    CONSFLG                     ; RESUME PRINTING ON CONTROL O
     50   0000000F'EF   9E   01A6   666         MOVAB   T_CNTRLO,R0                 ; ECHO STRING FOR CNTRLO
                0046   30   01AD   667         BSBW    OUT_CNTRL                   ; TYPE IT
00000000'EF   00010000 8F   CC   01B0   668         XORL2   #DS$M_CTRLO,DS$GL_FLAGS ; TOGGLE ^O FLAG
                                01BB   669
                     50   7C   01BB   670 80$:    CLRQ    R0                          ; Set to no character found
                                01BD   671
                          02   C1BD   672 90$:    REI                                 ; Return from interrupt
```

```
                        .SBTTL  RType Routine - Retype input line ('^R' function)
                ;++
                ; Functional description:
                ;
                ;       Cause a linefeed-carriage return, echo prompt and whatever characters
                ;       have been read so far from the terminal on the current line.
                ;
                ; Implicit inputs:
                ;
                ;       R2              count of characters input
                ;       QIO$_P5(AP)     prompt length/pointer from $QIO argument list
                ;       QIO$_P1(AP)     pointer to input buffer
                ;
                ; Side effects:
                ;
                ;       If this routine is interrupted, and output to terminal occurs, the
                ;       routine will restart itself--repeatedly if necessary, until the line
                ;       is output completely.
                ;--
```

```
                38   BB  01BE    694 RTYPE:  PUSHR   #^M<R3,R4,R5>              ; Save registers
55   00000000'EF    9E  01C0    695         MOVAB   L^DS$GL_FLAGS, R5          ; Save pointer to flags longword
                                01C7    696
50   00000014'EF    9E  01C7    697 10$:    MOVAB   T_CRLF, R0                 ; Type <CR><LF>
                4A  10  01CE    698         BSBB    OUT_ASCIC
     53   2C AC    7D  01D0    699         MOVQ    @QIO$_P5(AP), R3           ; Get prompt descriptor (ADDRESS/LENGTH)
                                01D4    700
        51   83    9A  01D4    701 20$:    MOVZBL  (R3)+, R1                  ; Get next character of prompt
     EC 65   17    E4  01D7    702         BBSC    S^#DS$V_OUTPUT,(R5),10$    ; Re-start if asynch typeout
                52  10  01DB    703         BSBB    OUT_CHAR                   ; Type it out
          F4 54    F5  01DD    704         SOBGTR  R4, 20$                    ; Loop to end of prompt
     54   1C BC    9E  01E0    705         MOVAB   @QIO$_P1(AP), R4           ; Point to buffer
        53   52    D0  01E4    706         MOVL    R2, R3                     ; Copy length
                                01E7    707
        51   84    9A  C1E7    708 30$:    MOVZBL  (R4)+, R1                  ; Get next character
     D9 65   17    E4  01EA    709         BBSC    S^#DS$V_OUTPUT,(R5),10$    ; Re-start if asynch typeout
                3F  10  01EE    710         BSBB    OUT_CHAR                   ; Type it
          F4 53    F5  01F0    711         SOBGTR  R3, 30$                    ; Loop to end of line
                38   BA  01F3    712         POPR    #^M<R3,R4,R5>              ; Restore registers
                05   01F5    713         RSB                                ; Return
```

```
                        715          .SBTTL  Out Routines - Flag driven terminal output
01F6    716 ;++
01F6    717 ; FUNCTIONAL DESCRIPTION:
01F6    718 ;
01F6    719 ;       OUT_CNTRL       Print out control character echo with low overhead
01F6    720 ;       OUT_ASCIC       OUTPUT ASCIC STRING R0=ADDRESS
01F6    721 ;       OUT_STRING      OUTPUT STRING R0=ADDRESS, R1=LENGTH
01F6    722 ;       CHAR_OUTPUT     OUTPUT CHARACTER R1=CHARACTER
01F6    723 ;       CHAR_ECHO       OUTPUT CHARACTER R1=CHARACTER, IGNORE ^O
01F6    724 ;
01F6    725 ; CALLING SEQUENCE:
01F6    726 ;
01F6    727 ;       NONE
01F6    728 ;
01F6    729 ; INPUT PARAMETERS:
01F6    730 ;
01F6    731 ;       NONE
01F6    732 ;
01F6    733 ; IMPLICIT INPUTS:
01F6    734 ;
01F6    735 ;       NONE
01F6    736 ;
01F6    737 ; OUTPUT PARAMETERS:
01F6    738 ;
01F6    739 ;       NONE
01F6    740 ;
01F6    741 ; IMPLICIT OUTPUTS:
01F6    742 ;
01F6    743 ;       NONE
01F6    744 ;
01F6    745 ; COMPLETION CODES:
01F6    746 ;
01F6    747 ;       NONE
01F6    748 ;
01F6    749 ; SIDE EFFECTS:         NONE
01F6    750 ;--
```

```
                                01F6   752 OUT_CNTRL:
                          52    DD 01F6   753         PUSHL   R2                      ; Save register 2
                    52    80    9A 01F8   754         MOVZBL  (R0)+,R2                ; Get length
                                   01FB   755
                    51    80    9A 01FB   756 10$:    MOVZBL  (R0)+,R1                ; Get character
                                   01FE   757         CMK     TCONSOLE                ; Type character
                      EB  52    F5 020D   758         SOBGTR  R2,10$                  ; Loop
              00000000'EF       94 0210   759         CLRB    COLUMN                  ; Control char. echo ends with <CR><LF>
                          52 8ED0 0216   760         POPL    R2                      ; Restore register
                             05 0219   761         RSB                             ; Return
                                0219   762
                                021A   763 OUT_ASCIC:
                    51    80    9A 021A   764         MOVZBL  (R0)+,R1                ; GET LENGTH
                                021D   765
                                C21D   766 OUT_STRING:
                          04    BB 021D   767         PUSHR   #^MR2                   ; SAVE
                    52    51    D0 021F   768         MOVL    R1,R2                   ; COPY LENGTH
                          08    15 0222   769         BLEQ    20$                     ; START WITH COUNT
                                0224   770
                    51    80    9A 0224   771 10$:    MOVZBL  (R0)+,R1                ; GET CHARACTER
                          06    10 0227   772         BSBB    OUT_CHAR                ; OUTPUT A CHARACTER
                      F8  52    F5 0229   773         SOBGTR  R2,10$                  ; NEXT
                                022C   774
                          04    BA 022C   775 20$:    POPR    #^MR2                   ; RESTORE
                                05 022E   776         RSB
                                022F   777
                                022F   778 OUT_CHAR:                            ; TELETYPE OUTPUT ROUTINE.
                       FE7A     30 022F   779         BSBW    KB_POLL                 ; Check for ^C and ^O
                                0232   780         Br_If_CtrlO     Out_CharX       ; Exit if ^O in effect                  [16]
                                023A   781
                                023A   782 OUT_ECHO:
                                023A   783         Br_If_Not_APT   5$              ; Branch around if not in APT mode      [16]
          0000FF00'EF     18    E0 0242   784         BBS     #DSA$V_NORPT, -
                             67 0246   785                   DSA$GL_FLAGS,OUT_CHARX; BRANCH IF REPORT DISABLED ?
                                024A   786
            05 00000003'EF       E9 024A   787 5$:     BLBC    CONSFLG,10$             ; BRANCH IF CONTROL S NOT ON
                       FE58     30 0251   788         BSBW    KB_POLL                 ; WAIT FOR CONTROL Q OR CONTROL C
                          F4    11 0254   789         BRB     5$
                                0256   790
                          51    0D 91 0256   791 10$:    CMPB    #CR,R1                  ; CARRIAGE RETURN ?
                                06 12 0259   792         BNEQ    20$                     ; NO -
                       00000000'EF       94 025B   793         CLRB    COLUMN                  ; INIT COLUMN COUNT
                                0261   794
00000002'EF     00000000'EF       91 0261   795 20$:    CMPB    COLUMN, DS$GB_WIDTH     ; Have we passed width limit?           [15]
                                0E 1F 026C   796         BLSSU   25$                     ; (Must be unsigned) No, continue       [12]
                                07 BB 026E   797         PUSHR   #^M<R0,R1,R2>           ; Save registers                        [12]
                    50  00000014'EF       9E 0270   798         MOVAB   T_CRLF, R0              ; Yes, Type CRLF, reset COLUMN          [12]
                             FFA0     30 0277   799         BSBW    OUT_ASCIC               ;                                       [12]
                                07 BA 027A   800         POPR    #^M<R0,R1,R2>           ; Restore                               [12]
                                027C   801
                                027C   802 25$:                                ;                                               [12]
                          51    09 91 027C   803         CMPB    #TAB,R1                 ; CHARACTER = TAB
                                0F 12 027F   804         BNEQ    50$                     ; NO -
                                51 20 9A 0281   805         MOVZBL  #SPACE,R1               ; SPACE
                                0284   806
                                0A 10 0284   807 30$:    BSBB    50$                     ; OUTPUT A SPACE
                00000000'EF     07 93 0286   808         BITB    #7,COLUMN               ; ALIGNED ON MULT OF 8 BOUNDRY?
```

```
                     F5   12  028D   809          BNEQ    30$                     ; NO, OUTPUT ANOTHER SPACE
                          05  028F   810          RSB                             ; YES, RETURN
                              0290   811
                  51  1F   91  0290   812 50$:     CMPB    #31,R1                  ; Control character?
                      06   18  0293   813          BGEQ    60$                     ; Not Horizontal motion if control char
          00000000'EF      96  0295   814          INCB    COLUMN                  ; Inc position
                              029B   815
00000001'EF   51      90  029B   816 60$:     MOVB    R1,LASTCHAR             ; Save character to be saved
                              02A2   817          CMK     TCONSOLE
                              02B1   818
                              02B1   819 OUT_CHARX:
                          05  02B1   820          RSB                             ; RETURN TO CALLING ROUTINE.
                              02B2   821
```

```
                                02B2    823 CMK$TCONSOLE::                              ; TELETYPE OUTPUT ROUTINE.
                          01    BB  02B2    824         PUSHR   #^MR0                   ; Save register
                                02B4    825
          22    00010000  8F    DA  02B4    826 10$:    MTPR    #^X10000,#PR$_TXCS      ; ACTIVATE CONSOLE TERMINAL LINE        [22]
                     50   22    DB  02BB    827         MFPR    #PR$_TXCS, R0           ; GET PRINTER STATUS.
                  F2 50   07    E1  02BE    828         BBC     #7, R0, 10$             ; WAIT FOR PRINTER READY BIT.
                     23   51    DA  02C2    829         MTPR    R1, #PR$_TXDB           ; OUTPUT CHARACTER.
                          01    BA  02C5    830         POPR    #^MR0                   ; Restore
                                02  02C7    831         REI                             ; RETURN TO CALLING ROUTINE.
```

```
                        833          .SBTTL  WriteVBlk Routine - Console write driver
                        834 ;++
                        835 ; FUNCTIONAL DESCRIPTION:
                        836 ;
                        837 ;       PERFORMS AN OUTPUT TO THE CONSOLE OF AN ASCII STRING.
                        838 ;
                        839 ; CALLING SEQUENCE:
                        840 ;
                        841 ;       CASE FROM QIO.
                        842 ;
                        843 ; INPUT PARAMETERS:
                        844 ;
                        845 ;       QIO$_EFN(AP)    = QIO EVENT FLAG NUMBER.
                        846 ;       QIO$_P1(AP)     = ADDRESS OF ASCII STRING.
                        847 ;       QIO$_P2(AP)     = LENGTH OF ASCII STRING.
                        848 ;
                        849 ; IMPLICIT INPUTS:
                        850 ;
                        851 ;       NONE
                        852 ;
                        853 ; OUTPUT PARAMETERS:
                        854 ;
                        855 ;       NONE
                        856 ;
                        857 ; IMPLICIT OUTPUTS:
                        858 ;
                        859 ;       NONE
                        860 ;
                        861 ; COMPLETION CODES:
                        862 ;
                        863 ;       NONE
                        864 ;
                        865 ; SIDE EFFECTS:
                        866 ;
                        867 ;       NONE
                        868 ;--
```

```
                        02C8   870 WRITEVBLK::
08 0C AC    00'  E1     02C8   871          BBC     S^#IO$V_CANCTRLO, -
                        02CD   872                  QIO$_FUNC(AP),10$       ; Branch if no Cancel Control O
                        02CD   873          Clear_CtrlO                     ; Clear the ^O flag                    [16]
                        02D5   874
   50   1C AC   7D      02D5   875 10$:     MOVQ    QIO$_P1(AP), R0         ; GET ADDRESS/LENGTH OF ASCII STRING.
          FF41  30      02D9   876          BSBW    OUT_STRING              ; OUTPUT THE STRING
                        02DC   877          $SETEF_G        (AP)
                  05    02E3   878          RSB                             ; RETURN TO ENTRY VECTOR.
```

```
                        880          .SBTTL  ReadVBlk Routine - Console read driver
                        881 ;++
                        882 ; FUNCTIONAL DESCRIPTION:
                        883 ;
                        884 ;       THE ROUTINE INPUTS CHARACTERS FROM THE TELETYPE AND STORES THEM
                        885 ;       IN THE BUFFER.  IT HANDLES ''DELETE'', ''^U'', ''^C'', ''^R'', AND
                        886 ;       ''HYPHEN_CARRTN''.
                        887 ;
                        888 ; CALLING SEQUENCE:
                        889 ;
                        890 ;       CASE FROM QIO.
                        891 ;
                        892 ; INPUT PARAMETERS:
                        893 ;
                        894 ;       QIO$_FUNC(AP)   = FUNCTION EITHER IO$_READVBLK OR IO$_READPROMPT
                        895 ;       QIO$_P1(AP)     = ADDRESS OF THE BEGINNING OF THE BUFFER.
                        896 ;       QIO$_IOSB(AP)   = ADDRESS TO HOLD LENGTH OF INPUT LINE
                        897 ;       QIO$_P2(AP)     = LENGTH IN BYTES ALLOWED FOR THE BUFFER.
                        898 ;       QIO$_P4(AP)     = OPTIONAL ADDR. OF TERMINATOR DESCRIPTOR.      [17]
                        899 ;
                        900 ; IMPLICIT INPUTS:
                        901 ;
                        902 ;       NONE
                        903 ;
                        904 ; OUTPUT PARAMETERS:
                        905 ;
                        906 ;       R1 = LENGTH OF LINE INPUT
                        907 ;       SUCCESS INDICATOR IN R0, R0 = 0 IF '?' TYPED ELSE R0 = 1
                        908 ;
                        909 ; IMPLICIT OUTPUTS:
                        910 ;
                        911 ;       NONE
                        912 ;
                        913 ; COMPLETION CODES:
                        914 ;
                        915 ;       NONE
                        916 ;
                        917 ; SIDE EFFECTS:
                        918 ;
                        919 ;       NONE
                        920 ;--
```

```
                                02E4    922 PREADVBLK::
                                02E4    923 READVBLK::
00000000'EF   00810000 8F   CA  02E4    924          BICL2   #DS$M_CTRLO ! DS$M_OUTPUT, -
                                02EF    925                   DS$GL_FLAGS           ; Clear ^O and output flag
            0C AC   00'8F   91  02EF    926          CMPB    #IO$_READPROMPT,QIO$_FUNC(AP) ; READ W/PROMPT?
                       22   12  02F4    927          BNEQ    10$                   ; Branch if not
              00000000'EF   95  02F6    928          TSTB    COLUMN                ; At left margin?
                       13   12  02FC    929          BNEQ    6$                    ; Branch if not
          00000001'EF   0A   91  02FE    930          CMPB    #LF,LASTCHAR          ; Was last char a LF?
                       0A   13  0305    931          BEQL    6$                    ; Branch if so
                                0307    932
          50   00000014'EF   9E  0307    933 3$:      MOVAB   T_CRLF,R0             ; Output CRLF
                     FF09   30  030E    934          BSBW    OUT_ASCIC             ; Type it
                                0311    935
               50   2C AC   7D  0311    936 6$:      MOVQ    QIO$_P5(AP),R0        ; GET ADDRESS,LENGTH OF PROMPT
                     FF05   30  0315    937          BSBW    OUT_STRING            ; AND TYPE IT
                                0318    938
               53   1C BC   9E  0318    939 10$:     MOVAB   @QIO$_P1(AP), R3      ; SET BUFFER POINTER IN R3
                          52   D4  031C    940          CLRL    R2                    ; INITIALIZE BYTE COUNTER IN R2
                                031E    941
                     FDBF   30  031E    942 20$:     BSBW    RINPUT                ; GO GET A CHARACTER
                                0321    943          Clear_CtrlO           ; Clear the ^O flag                    [16]
               50   80 8F   8A  0329    944          BICB2   #PARITY, R0           ; CLEAR PARITY BIT
               50   7F 8F   91  032D    945          CMPB    #DELETE, R0           ; IS THIS "DELETE"
                          20   12  0331    946          BNEQU   50$                   ; BRANCH IF NOT
                          52   D5  0333    947          TSTL    R2                    ; IS THIS BEGINNING OF BUFFER
                          03   12  0335    948          BNEQ    30$                   ; BRANCH IF NOT BEGINNING
                        00E4   31  0337    949          BRW     140$                  ; GO TYPE BELL IF BEGINNING
                                033A    950
          00000000'EF   05   E2  033A    951 30$:     BBSS    #DS$V_RUBFLG, -       ; BRANCH IF BSLASH ALREADY TYPED
                          07           0341    952                   DS$GL_FLAGS, 40$
               51   5C 8F   9A  0342    953          MOVZBL  #BSLASH, R1           ; LOAD BACK SLASH INTO R1
                     FEE6   30  0346    954          BSBW    OUT_CHAR              ; TYPE BACK SLASH
                                0349    955
                  51   73   9A  0349    956 40$:     MOVZBL  -(R3), R1             ; LOAD DELETED CHARACTER INTO R1
                     FEE0   30  034C    957          BSBW    OUT_CHAR              ; TYPE DELETED CHARACTER
                          52   D7  034F    958          DECL    R2                    ; ADJUST CHARACTER COUNTER
                          CB   11  0351    959          BRB     20$                   ; GO GET ANOTHER CHARACTER
                                0353    960
          00000000'EF   05   E5  0353    961 50$:     BBCC    #DS$V_RUBFLG, -       ; Branch if last char not "DEL"
                          07           035A    962                   DS$GL_FLAGS, 60$
               51   5C 8F   9A  035B    963          MOVZBL  #BSLASH, R1           ; LOAD BACK SLASH INTO R1
                     FECD   30  035F    964          BSBW    OUT_CHAR              ; TYPE BACK SLASH
                                0362    965
                  50   15   91  0362    966 60$:     CMPB    #CNTRLU, R0           ; IS THIS CONTROL U
                          0D   12  0365    967          BNEQU   70$                   ; BRANCH IF NOT
    50    00000007'EF   9E  0367    968          MOVAB   L^T_CNTRLU,R0         ; ADDRESS OF TEXT                      [14]
                     FEA9   30  036E    969          BSBW    OUT_ASCIC             ; TYPE IT
                     FF70   31  0371    970          BRW     PREADVBLK             ; GO INITIALIZE BUFFER
                                0374    971
                  50   03   91  0374    972 70$:     CMPB    #CNTRLC, R0           ; IS THIS CONTROL C
                          0D   12  0377    973          BNEQU   90$                   ; BRANCH IF NOT
                  53   50   9A  0379    974          MOVZBL  R0, R3                ; Set terminator character
                          52   D4  037C    975          CLRL    R2                    ; FLUSH BUFFER.
        50   0000'8F   3C  037E    976          MOVZWL  #SS$_CONTROLC,R0      ; SET RETURN CODE
                     00AF   31  0383    977          BRW     170$
                                0386    978
```

```
                50   12   91   0386   979 90$:     CMPB    #CNTRLR, R0             ; IS THIS CONTROL R
                     17   12   0389   980          BNEQU   110$                    ; BRANCH IF NOT
                     52   D5   038B   981          TSTL    R2                      ; IS THIS BEGINNING OF BUFFER
                     03   12   038D   982          BNEQU   100$                    ; NO                                    [17]
                   008C   31   038F   983          BRW     140$                    ; GO TYPE BELL IF BEGINNING             [17]
                                0392   984 100$:                                   ;                                       [17]
          50  0000000C'EF   9E   0392   985          MOVAB   L^T_CNTRLR,R0           ; OUTPUT ''^R<CR>''                             [14]
                   FE7E   30   0399   986          BSBW    OUT_ASCIC
                   FE1F   30   039C   987          BSBW    RTYPE                   ; Re-type input line
                   FF7C   31   039F   988          BRW     20$                     ; BRANCH TO GET NEXT CHAR
                                03A2   989
                                03A2   990 110$:
                                03A2   991 ;[23]     CMPB    #QUESTN, R0             ; IS THIS A QUESTION MARK
                                03A2   992 ;[23]     BNEQU   120$                    ; BRANCH IF NOT
                                03A2   993 ;[23]     MOVZBL  R0, R1                  ; OUTPUT QUESTION MARK
                                03A2   994 ;[23]     BSBW    OUT_CHAR                ; TYPE QUESTION MARk
                                03A2   995 ;[23]     MOVZBL  R0, R3                  ; Set terminator character
                                03A2   996 ;[23]     MOVL    #CR,R1                  ; RETURN THE CARRIAGE
                                03A2   997 ;[23]     BSBW    OUT_CHAR
                                03A2   998 ;[23]     CLRL    R0                      ; SET ERROR RETURN
                                03A2   999 ;[23]     BRW     170$                    ; BRANCH TO EXIT                        [17]
                                03A2  1000
                50   0D   91   03A2  1001 120$:     CMPB    #CR, R0                 ; IS THIS CARRIAGE RETURN
                     17   12   03A5  1002          BNEQU   130$                    ; BRANCH IF NOT
                63   50   90   03A7  1003          MOVB    R0,(R3)                 ; STORE THE CHARACTER... MAY BE IGNORED
                51   50   D0   03AA  1004          MOVL    R0,R1                   ; COPY FOR ECHO
                   FE7F   30   03AD  1005          BSBW    OUT_CHAR                ; ECHO <CR>
                51   0A   D0   03B0  1006          MOVL    #LF,R1
                   FE79   30   03B3  1007          BSBW    OUT_CHAR                ; TYPE <LF>
                50   01   D0   03B6  1008          MOVL    #1,R0                   ; ASSUME IT'S <CR> AND SET SUCCESS
                53   0D   9A   03B9  1009          MOVZBL  #CR, R3                 ; Set terminator character
                     77   11   03BC  1010          BRB     170$                    ; Exit
                                03BE  1011
                                03BE  1012 130$:                                 ;                                       [17]
          54    28   AC   D0   03BE  1013          MOVL    Q10$_P4(AP),R4          ;IF TERMINATOR DESCRIPTOR PASSED        [17]
                     28   13   03C2  1014          BEQL    4000$                   ;THEN                                   [17]
                     64   D5   03C4  1015          TSTL    (R4)                    ;   IF SHORT FORM                       [17]
                     08   12   03C6  1016          BNEQ    1000$                   ;   THEN (SEE VMS I/O USERS GDE)        [17]
                56   1F   D0   03C8  1017          MOVL    #31,R6                  ;      INIT BIT POINTER TO MAX          [17]
                54   04   C0   03CB  1018          ADDL2   #4,R4                   ;      GET ADDR. OF BIT PATTERN         [17]
                     0C   11   03CE  1019          BRB     2000$                   ;   ELSE                                [17]
                56   64   D0   03D0  1020 1000$:    MOVL    (R4),R6                 ;      GET BYTE COUNT OF BIT PAT        [17]
                56   08   C4   03D3  1021          MULL2   #8,R6                   ;      CONVERT TO BIT POINTER           [17]
                54   04   C0   03D6  1022          ADDL2   #4,R4                   ;      GET ADDR. OF BIT PAT. PTR        [17]
                54   64   D0   03D9  1023          MOVL    (R4),R4                 ;      GET ADDR. OF BIT PATTERN         [17]
                                03DC  1024 2000$:                                ;   REPEAT
                64   56   E1   03DC  1025          BBC     R6,(R4),-               ;      IF THIS BIT IS SET               [17]
                     05        03DF  1026                  3000$                   ;      THEN                             [17]
                56   50   91   03E0  1027          CMPB    R0,R6                   ;         IF BIT NO. = INP. CHR ASCII   [17]
                     25   13   03E3  1028          BEQL    5000$                   ;         THEN LEAVE LOOP               [17]
                                03E5  1029 3000$:                                ;      ELSE                             [17]
                      6   D7   03E5  1030          DECL    R6                      ;         DECR. BIT POINTER             [17]
                     F3   18   03E7  1031          BGEQ    2000$                   ;   UNTIL ENTIRE BIT PAT. DONE           [17]
                   002C   31   03E9  1032          BRW     6000$                   ;   CHR NOT TERMINATOR, SO CONT.         [17]
                                03EC  1033 4000$:                                ;ELSE                                   [17]
                50   1F   91   03EC  1034          CMPB    #31,R0                  ;   IF INP. CHR ASCII LSS 31            [17]
                     27   1B   03EF  1035          BLEQU   6000$                   ;   THEN NOT TERMINATOR                 [17]
```

```
                50   0A   91   03F1  1036          CMPB    #LF,R0                  ;    IF INP. CHR IS LINEFEED      [17]
                     22   13   03F4  1037          BEQL    6000$                   ;    THEN NOT TERMINATOR          [17]
                50   0B   91   03F6  1038          CMPB    #VT,R0                  ;    IF INP. CHR IS VT            [17]
                     1D   13   03F9  1039          BEQL    6000$                   ;    THEN NOT TERMINATOR          [17]
                50   0C   91   03FB  1040          CMPB    #FF,R0                  ;    IF INP. CHR IS FORMFEED      [17]
                     18   13   03FE  1041          BEQL    6000$                   ;    THEN NOT TERMINATOR          [17]
                50   09   91   0400  1042          CMPB    #TAB,R0                 ;    IF INP. CHR IS TAB           [17]
                     13   13   0403  1043          BEQL    6000$                   ;    THEN NOT TERMINATOR          [17]
                50   08   91   0405  1044          CMPB    #BS,R0                  ;    IF INP. CHR IS BACKSPACE     [17]
                     0E   13   0408  1045          BEQL    6000$                   ;    THEN NOT TERMINATOR          [17]
                          040A  1046 5000$:                                       ;TERMINATOR HAS BEEN FOUND        [17]
                51   50   D0   040A  1047          MOVL    R0,R1                   ;PUT CHAR INTO R1                 [17]
                   FE1F   30   040D  1048          BSBW    OUT_CHAR                ;PRINT IT                         [17]
                83   50   D0   0410  1049          MOVL    R0,(R3)+                ;STORE IT                         [17]
                     52   D6   0413  1050          INCL    R2                      ;COUNT IT                         [17]
                   001D   31   0415  1051          BRW     170$                    ;EXIT                             [17]
                          0418  1052 6000$:                                       ;                                 [17]
            20  AC   52   D1   0418  1053          CMPL    R2, QIO$_P2(AP)         ; IS THIS LAST BYTE IN BUFFER
                     09   19   041C  1054          BLSS    150$                    ; BRANCH IF NOT LAST CHARACTER
                          041E  1055
                51   07   9A   041E  1056 140$:    MOVZBL  #BELL, R1               ; LOAD 'BELL' CODE INTO R1
                   FE0B   30   0421  1057          BSBW    OUT_CHAR                ; TYPE 'BELL'
                   FEF7   31   0424  1058          BRW     20$                     ; TRY FOR SOMETHING ELSE
                          0427  1059
                51   50   9A   0427  1060 150$:    MOVZBL  R0, R1                  ; LOAD CHARACTER INTO R1
                   FE02   30   042A  1061          BSBW    OUT_CHAR                ; ECHO CHARACTER
                83   50   90   042D  1062          MOVB    R0, (R3)+               ; STORE CHARACTER IN BUFFER
                     52   D6   0430  1063          INCL    R2                      ; COUNT THIS CHARACTER
                   FEE9   31   0432  1064          BRW     20$                     ; GO GET ANOTHER CHARACTER
                          0435  1065
            51    10  BC   DE   0435  1066 170$:    MOVAL   @QIO$_IOSB(AP), R1      ; GET ADDRESS OF I/O STATUS BLOCK.
                81   50   B0   0439  1067          MOVW    R0, (R1)+               ; STORE OPERATION STATUS
                81   52   B0   043C  1068          MOVW    R2, (R1)+               ; Store character count.
                81   53   B0   043F  1069          MOVW    R3, (R1)+               ; Store terminator char.
                61   01   B0   0442  1070          MOVW    S^#1, (R1)              ; Store terminator size
                          0445  1071          $SETEF_G (AP)                       ; SET EVENT FLAG.
                          05   044C  1072          RSB                             ; RETURN TO QIO ENTRY VECTOR.
                               044D  1073
```

```
                        1075 .sbttl SIZE_TERM Routine to size the console terminal
                        1076
                        1077 ;++
                        1078 ; FUNCTIONAL DESCRIPTION:                                                          [23]
                        1079 ;                                                                                  [23]
                        1080 ;       This routine will size the console terminal to determine its               [23]
                        1081 ;       characteristics.  The routine is meant for standalone use,                 [23]
                        1082 ;       and attempts to provide some of the same information that                  [23]
                        1083 ;       can be obtained for timesharing terminals in user mode                     [23]
                        1084 ;       using the $GETCHN service.  The routine sets the "type"                    [23]
                        1085 ;       and "terminal characteristics" fields of DS$GL_TERMCHAR.                   [23]
                        1086 ;       The fields are defined the the $TTDEF macro of VMS.                        [23]
                        1087 ;                                                                                  [23]
                        1088 ;       The format of DS$GL_TERMCHAR is as follows:                                [23]
                        1089 ;                                                                                  [23]
                        1090 ;       32                                  0                                      [23]
                        1091 ;       .....................................                                      [23]
                        1092 ;       : page                :       :     :                                      [23]
                        1093 ;       :   width             : type  : class :                                    [23]
                        1094 ;       :.....................:.......:.......:                                    [23]
                        1095 ;       :page   :    terminal                 :                                    [23]
                        1096 ;       : length:     characteristics         :                                    [23]
                        1097 ;       :.......:.............................:                                    [23]
                        1098 ;                                                                                  [23]
                        1099 ;       NOTE: Currently, the routine only sets the TT$M_SCOPE                      [23]
                        1100 ;       characteristic.  Other characteristics defined by $TTDEF,                  [23]
                        1101 ;       or other fields in DS$GL_TERMCHAR, could be also be sized for.             [23]
                        1102 ;                                                                                  [23]
                        1103 ;                                                                                  [23]
                        1104 ; CALLING SEQUENCE:                                                                [23]
                        1105 ;                                                                                  [23]
                        1106 ;       BSBW    SIZE_TERM                                                          [23]
                        1107 ;                                                                                  [23]
                        1108 ; INPUT PARAMETERS:                                                                [23]
                        1109 ;                                                                                  [23]
                        1110 ;       NONE                                                                       [23]
                        1111 ;                                                                                  [23]
                        1112 ; IMPLICIT INPUTS:                                                                 [23]
                        1113 ;                                                                                  [23]
                        1114 ;       DS$GL_TERMCHAR                                                             [23]
                        1115 ;                                                                                  [23]
                        1116 ; OUTPUT PARAMETERS:                                                               [23]
                        1117 ;                                                                                  [23]
                        1118 ;       NONE                                                                       [23]
                        1119 ;                                                                                  [23]
                        1120 ; IMPLICIT OUTPUTS:                                                                [23]
                        1121 ;                                                                                  [23]
                        1122 ;       DS$GL_TERMCHAR                                                             [23]
                        1123 ;                                                                                  [23]
                        1124 ; COMPLETION CODES:                                                                [23]
                        1125 ;                                                                                  [23]
                        1126 ;                                                                                  [23]
                        1127 ;                                                                                  [23]
                        1128 ; SIDE EFFECTS:                                                                    [23]
                        1129 ;                                                                                  [23]
                        1130 ;                                                                                  [23]
                        1131 ;                                                                                  [23]
```

```
                                044D  1132 ;--                                                                    [23]
                                044D  1133
                     5A 1B 00'  044D  1134 DEC_PROMPT:     .ASCIC  <27><90>         ;DEC prompt                 [23]
                           02   044D
                  63 5B 1B 00'  0450  1135 ANSI_PROMPT:    .ASCIC  <27>'[c'         ;ANSI prompt                [23]
                           03   0450
                        05 00'  0454  1136 LA36_PROMPT:    .ASCIC  <5>              ;LA36 prompt                [23]
                           01   0454
                                0456  1137
                                0456  1138 PROMPTS:                                ; Pointers to prompt strings [23]
                    0000044D'   0456  1139         .ADDRESS DEC_PROMPT
                    00000450'   045A  1140         .ADDRESS ANSI_PROMPT
                    00000454'   045E  1141         .ADDRESS LA36_PROMPT
                    00000000    0462  1142         .LONG   0
                                C466  1143
                    00000014    0466  1144 RECVD_BUF_LENGTH = 20                  ; (Longest response string will [23]
                                0466  1145                                        ;  probably be from LA36, which [23]
                                0466  1146                                        ;  can be up to 20 chars. But we[23]
                                0466  1147                                        ;  don't know what LA36 string  [23]
                                0466  1148                                        ;  is anyway;it is user-defined)[23]
                    0000047A    0466  1149 RECVD_BUF:      .BLKB RECVD_BUF_LENGTH  ; Buffer to receive response  [23]
2E 31 3A 3A 20 30 00000482'010E0000'  047A  1150 MAX_WAIT:       .ASCID  /0 ::1.5/       ; Max. time to wait for response[23]
                           35   0488
                    00000491    0489  1151 QUAD_TIME:      .BLKQ   1               ; Converted binary time      [23]
                                0491  1152
                        01FC    0491  1153 .ENTRY SIZE_TERM, ^M<R2,R3,R4,R5,R6,R7,R8>;                           [23]
                                0493  1154
                                0493  1155         $BINTIM_S MAX_WAIT, QUAD_TIME   ; Convert ASCII time to binary [23]
              00000000'EF   7C  04A0  1156         CLRQ    DS$GL_TERMCHAR          ; Initialize DS$GL_TERMCHAR  [23]
               57   AD AF   9E  04A6  1157         MOVAB   PROMPTS, R7             ; Get list of prompts        [23]
                  58   87   D0  04AA  1158 10$:    MOVL    (R7)+,R8                ; Get address of next prompt [23]
                       03   12  04AD  1159         BNEQ    15$                     ; Any more remaining?        [23]
                     0084   31  04AF  1160         BRW     40$                     ; Branch if no more prompts  [23]
                            04B2  1161 15$:    $SETIMR_S       #1,QUAD_TIME    ; Start timer.  (This MUST be [23]
                            04C0  1162                                         ;  done before the prompt is [23]
                            04C0  1163                                         ;  sent, because the $SETIMR  [23]
                            04C0  1164                                         ;  service calls KB_CHECK, which[23]
                            04C0  1165                                         ;  will read KB input and throw [23]
                            04C0  1166                                         ;  it away.)                   [23]
                  50   58   D0  04C0  1167         MOVL    R8,R0                   ; Put prompt addr. in R0     [23]
                     FD30   30  04C3  1168         BSBW    OUT_CNTRL               ; Send prompt to terminal    [23]
                    9D AF   94  04C6  1169         CLRB    RECVD_BUF               ; Clear 1st byte of response buf[23]
               58   9A AF   9E  04C9  1170         MOVAB   RECVD_BUF,R8            ; Init response buffer pointer [23]
                            04CD  1171 20$:                                        ; REPEAT
                            04CD  1172 30$:    $READEF_S       #1,EVENT_FLAGS  ; See if time was used up    [23]
50  00000000'8F   D1  04DC  1173         CMPL    #SS$_WASSET,R0          ; Has time run out?          [23]
                       18   13  04E3  1174         BEQL    35$             ; Yes.  No response.         [23]
                            04E5  1175         CMK     RCONSOLE        ; Get a character of response [23]
                  D5 50   07   E1  04F4  1176         BBC     #7,R0,30$       ; If no character, try again [23]
                     88   51   90  04F8  1177         MOVB    R1,(R8)+        ; Store char. in response buffer[23]
                          D0   11  04FB  1178         BRB     30$             ; Get another character      [23]
                     FF65 CF   95  04FD  1179 35$:    TSTB    RECVD_BUF       ; IF no characters were received[23]
                          A7   13  0501  1180         BEQL    10$             ; THEN try another prompt     [23]
                               0503  1181
                               0503  1182 ;
                               0503  1183 ; A response was received.  Now match the response to a terminal type.  [23]
                               0503  1184 ;
```

```
                        0503  1185
                        0503  1186          MOVL    #TT$M_SCOPE, R8                 ; Assume video terminal          [23]
                        050A  1187          MOVAB   VIDEO_TYPES, R4                 ; Point to video response list   [23]
                        0511  1188          BSBW    100$                            ; Search list                    [23]
                        0514  1189          BLBS    R0,40$                          ; Found a match                  [23]
                        0517  1190          CLRL    R8                              ; Not a video terminal           [23]
                        0519  1191          MOVAB   HARDCOPY_TYPES, R4              ; Point to hardcopy response list[23]
                        0520  1192          BSBW    100$                            ; Search list                    [23]
                        0523  1193          BLBS    R0,40$                          ; Found a match                  [23]
00000000'EF  08  08     0526  1194          INSV    #TT$_LA36,#8,#8, -              ; No match, so chances are that  [23]
                        052F  1195                  DS$GL_TERMCHAR                  ;  the terminal is an LA36, since[23]
00000004'EF  58         052F  1196          MOVL    R8,DS$GL_TERMCHAR+4             ;  they do not return a standard[23]
                        0536  1197                                                  ;  response.                     [23]
                        0536  1198 40$:      RET                                     ; BYE.                           [23]
                        0537  1199
                        0537  1200 100$:     ;This code searches a list of terminal responses and compares    [23]
                        0537  1201          ;them to the received response, looking for a match.  If a        [23]
                        0537  1202          ;match is found, DS$GL_TERMCHAR is set up appropriately.          [23]
                        0537  1203
50                      0537  1204          CLRL    R0                              ; Clear return status            [23]
FF 8F                   0539  1205          CMPB    (R4), #-1                       ; End of list?                   [23]
28                      053D  1206          BEQL    300$                            ; Yes                            [23]
55                      053F  1207          MOVZBL  (R4)+, R5                       ; Get length                     [23]
FF1E CF                 0542  1208          CMPC3   R5,(R4),RECVD_BUF               ; Compare strings                [23]
07                      0548  1209          BEQL    200$                            ; Found a match                  [23]
54                      054A  1210          ADDL    R5,R4                           ; Go to end of string            [23]
54                      054D  1211          INCL    R4                              ; Skip over type code            [23]
E6                      054F  1212          BRB     100$                            ; Try next response in list      [23]
54                      0551  1213 200$:     ADDL    R5,R4                           ; Point to type code             [23]
00000000'EF  08  08     0554  1214          INSV    (R4),#8,#8,DS$GL_TERMCHAR       ; Store type code                [23]
00000004'EF  58         055D  1215          MOVL    R8,DS$GL_TERMCHAR+4             ; Store characteristics          [23]
50  01                  0564  1216          MOVL    #1,R0                           ; Set good return status         [23]
                        0567  1217 300$:     RSB                                     ; Leave                          [23]
                        0568  1218
                        0568  1219          .END
```

```
$$ARGS                          =  0000000C     D
$$N                             =  00000002     D
$$T1                            =  00000000     D
ANSI_PROMPT                        00000450 R   D   04
BELL                            =  00000007     D
BIT...                          =  00000004     D
BS                              =  00000008     D
BSLASH                          =  0000005C     D
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_SIZE                         00000444     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_LAST                         0000C024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEST                         00000020     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       0000003C     D
CLI$V_ADAPTER                   =  00000018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BREAK                     =  0000000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
CMK$RCONSOLE                       00000109 RG  D   04
CMK$TCONSOLE                       000002B2 RG  D   04
CMK$_                           =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
CNTRLC                          =  00000003     D
CNTRLO                          =  0000000F     D
CNTRLQ                          =  00000011     D
CNTRLR                          =  00000012     D
CNTRLS                          =  00000013     D
CNTRLU                          =  00000015     D
COLUMN                             00000000 R   D   02
CONSFLG                            00000003 R   D   02
CR                              =  0000000D     D
CRD$M_CRD_DEBUG                 =  00000001     D
CRD$M_CTRL_C_NO_ECHO            =  00000002     D
CRD$M_FLUSH_CTRL_C              =  00000004     D
CRD$M_INHIBIT_ALLOCATE          =  00000008     D
CRD$V_CRD_DEBUG                 =  00000000     D
CRD$V_CTRL_C_NO_ECHO            =  00000001     D
CRD$V_FLUSH_CTRL_C              =  00000002     D
CRD$V_INHIBIT_ALLOCATE          =  00000003     D
DEC_PROMPT                         0000044D R   D   04
DELETE                          =  0000007F     D
DS$GB_WIDTH                        00000002 RG  D   02
DS$GL_CRD_FLAGS                    ********     X   04
DS$GL_FLAGS                        ********     X   04
DS$GL_TERMCHAR                     ********     X   04
DS$GW_TTOUT                        ********     X   04
DS$K_PRINTB                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  0000001B     D
```

```
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        =  0000000B     D
DS$K_TYPE_FIRST_PASS            =  00000011     D
DS$K_TYPE_GENERAL               =  00000000     D
DS$K_TYPE_GENERAL_ERROR         =  00000003     D
DS$K_TYPE_NO_TESTS              =  00000012     D
DS$K_TYPE_PARAM_ERROR           =  0000001C     D
DS$K_TYPE_PROGRAM_END           =  00000010     D
DS$K_TYPE_PROGRAM_INFO          =  00000017     D
DS$K_TYPE_PROGRAM_START         =  0000000F     D
DS$K_TYPE_QIO_INVADP            =  00000024     D
DS$K_TYPE_QIO_NODRIVER          =  00000022     D
DS$K_TYPE_QIO_WRONGVER          =  00000023     D
DS$K_TYPE_SCRIPT_ECHO           =  00000021     D
DS$K_TYPE_SCRIPT_PNF            =  0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         =  00000020     D
DS$K_TYPE_SCRIPT_SKIP           =  0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        =  00000019     D
DS$K_TYPE_START_ERR             =  00000018     D
DS$K_TYPE_START_LIST            =  00000025     D
DS$K_TYPE_SUMMARY               =  0000000E     D
DS$K_TYPE_USER_PROMPT           =  00000002     D
DS$M_ABRTFLG                    =  00000040     D
DS$M_BADTIME                    =  00100000     D
DS$M_BATCH                      =  00400000     D
DS$M_BRKCLR                     =  00001000     D
DS$M_BRKPT                      =  00000800     D
DS$M_CHARFLG                    =  00000100     D
DS$M_CMDFLG                     =  00000080     D
DS$M_CTRLC                      =  00000001     D
DS$M_CTRLO                      =  00010000     D
DS$M_DEVFLG                     =  00000200     D
DS$M_DISABLCC                   =  01000000     D
DS$M_DONFLG                     =  00002000     D
DS$M_ERRFLG                     =  00000010     D
DS$M_EXCEPT                     =  00080000     D
DS$M_EXETST                     =  00040000     D
DS$M_HLTFLG                     =  00000008     D
DS$M_LODFLG                     =  00000002     D
DS$M_MEMMGT                     =  00008000     D
DS$M_OUTPUT                     =  00800000     D
DS$M_RUBFLG                     =  00000020     D
DS$M_SCRIPT                     =  00200000     D
DS$M_SETIMR                     =  02000000     D
DS$M_STRFLG                     =  00000004     D
DS$M_SUBT                       =  00004000     D
DS$M_SYSFLG                     =  00000400     D
DS$M_TIMRON                     =  00020000     D
DS$V_ABRTFLG                    =  00000006     D
DS$V_BADTIME                    =  00000014     D
DS$V_BATCH                      =  00000016     D
DS$V_BRKCLR                     =  0000000C     D
DS$V_BRKPT                      =  0000000B     D
DS$V_CHARFLG                    =  00000008     D
DS$V_CMDFLG                     =  00000007     D
DS$V_CTRLC                      =  00000000     D
DS$V_CTRLO                      =  00000010     D
DS$V_DEVFLG                     =  00000009     D
DS$V_DISABLCC                   =  00000018     D
DS$V_DONFLG                     =  0000000D     D
DS$V_ERRFLG                     =  00000004     D
DS$V_EXCEPT                     =  00000013     D
DS$V_EXETST                     =  00000012     D
DS$V_HLTFLG                     =  00000003     D
DS$V_LODFLG                     =  00000001     D
DS$V_MEMMGT                     =  0000000F     D
DS$V_OUTPUT                     =  00000017     D
DS$V_RUBFLG                     =  00000005     D
DS$V_SCRIPT                     =  00000015     D
DS$V_SETIMR                     =  00000019     D
DS$V_STRFLG                     =  00000002     D
DS$V_SUBT                       =  0000000E     D
DS$V_SYSFLG                     =  0000000A     D
DS$V_TIMRON                     =  00000011     D
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSA$V_APT                       =  0000001F     D
DSA$V_NORPT                     =  0000001B     D
DSA$V_USER                      =  0000001C     D
DSX$PRINT                          ********     X    04
END_HARDCOPY_TYPES                 00000138 R   D    03
END_VIDEO_TYPES                    0000010B R   D    03
EVENT_FLAGS                        00000004 R   D    02
FF                              =  0000000C     D
HARDCOPY_TYPES                     0000010F R   D    03
HYPHEN                          =  0000002D     D
IO$V_CANCTRLO                      ********     X    04
IO$_READPROMPT                     ********     X    04
IO$_SENSEMODE                      ********     X    04
IO$_SETMODE                        ********     X    04
KB_POLL                            000000AC RG  D    04
KB_POLL_X                          000000DD R   D    04
LA36_PROMPT                        00000454 R   D    04
LASTCHAR                           00000001 R   D    02
LF                              =  0000000A     D
```

```
LOWCASE                         =  00000020     D
MAX_WAIT                           0000047A R   D   04
OUT_ASCIC                          0000021A R   D   04
OUT_CHAR                           0000022F R   D   04
OUT_CHARX                          000002B1 R   D   04
OUT_CNTRL                          000001F6 R   D   04
OUT_ECHO                           0000023A R   D   04
OUT_STRING                         0000021D R   D   04
PARITY                          =  00000080     D
PR$_RXCS                        =  00000020     D
PR$_RXDB                        =  00000021     D
PR$_TXCS                        =  00000022     D
PR$_TXDB                        =  00000023     D
PREADVBLK                          000002E4 RG  D   04
PROMPTS                            00000456 R   D   04
PSL$V_IS                        =  0000001A     D
QIO$_ASTADR                     =  00000014     D
QIO$_ASTPRM                     =  00000018     D
QIO$_CHAN                       =  00000008     D
QIO$_EFN                        =  00000004     D
QIO$_FUNC                       =  0000000C     D
QIO$_IOSB                       =  00000010     D
QIO$_NARGS                      =  0000000C     D
QIO$_P1                         =  0000001C     D
QIO$_P2                         =  00000020     D
QIO$_P3                         =  00000024     D
QIO$_P4                         =  00000028     D
QIO$_P5                         =  0000002C     D
QIO$_P6                         =  00000030     D
QUAD_TIME                          00000489 R   D   04
QUESTN                          =  0000003F     D
READVBLK                           000002E4 RG  D   04
RECVD_BUF                          00000466 R   D   04
RECVD_BUF_LENGTH                =  00000014     D
RINPUT                             000000E0 R   D   04
RINPUTX                            00000108 R   D   04
RTYPE                              000001BE R   D   04
SIZ...                          =  00000001     D
SIZE_TERM                          00000491 RG  D   04
SPACE                           =  00000020     D
SS$_CONTROLC                       ********   X   04
SS$_WASSET                         ********   X   04
SYS$BINTIM                         ********  GX   04
SYS$QIOW                           ********  GX   04
SYS$READEF                         ********  GX   04
SYS$SETEF                          ********  GX   04
SYS$SETIMR                         ********  GX   04
TAB                             =  00000009     D
TT$M_SCOPE                      =  00001000     D
TT$_LA100                       =  00000025     D
TT$_LA12                        =  00000024     D
TT$_LA120                       =  00000021     D
TT$_LA34                        =  00000022     D
TT$_LA36                        =  00000020     D
TT$_UNKNOWN                     =  00000000     D
TT$_VT100                       =  00000060     D
TT$_VT52                        =  00000040     D
TT$_VT55                        =  00000041     D
TT$_VT5X                        =  00000040     D
T_CNTRLC                           00000000 R   D   03
T_CNTRLO                           0000000F R   D   03
T_CNTRLR                           0000000C R   D   03
T_CNTRLU                           00000007 R   D   03
T_CRLF                             00000014 R   D   03
T_LA100                            00000117 R   D   03
T_LA12                             0000010F R   D   03
T_LA120                            0000011F R   D   03
T_LA34                             00000126 R   D   03
T_LA34A                            0000012F R   D   03
T_LA36                             0000013C R   D   03
T_SHOW_WIDTH                       00000048 R   D   03
T_VT100                            0000009F R   D   03
T_VT100A                           0000009A R   D   03
T_VT100B                           000000A8 R   D   03
T_VT100C                           000000B1 R   D   03
T_VT100D                           000000BA R   D   03
T_VT100E                           000000C3 R   D   03
T_VT100F                           000000CC R   D   03
T_VT100G                           000000D5 R   D   03
T_VT100H                           000000DE R   D   03
T_VT220                            000000E7 R   D   03
T_VT240                            000000F7 R   D   03
T_VT50_A                           0000006D R   D   03
T_VT50_B                           00000072 R   D   03
T_VT50_C                           00000077 R   D   03
T_VT50_H                           00000081 R   D   03
T_VT50_J                           00000086 R   D   03
T_VT52_K                           0000008B R   D   03
T_VT52_L                           00000090 R   D   03
T_VT52_M                           00000095 R   D   03
T_VT55_E                           0000007C R   D   03
T_WIDTH                            00000017 R   D   03
VIDEO_TYPES                        0000006D R   D   03
VRSETWIDTH                         00000000 RG  D   04
VRSHOWWIDTH                        0000008C RG  D   04
VT                              =  0000000B     D
WRITEVBLK                          000002C8 RG  D   04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	0000013D (317.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000568 (1384.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=0000000C	159 (1)	159 (1)
$$N	=00000002	490 (1)	#-442 (1) 490 (1)
$$T1	=00000000	1161 (1)	1161 (1) 159 (1) 428 (1) 435 (1)
ANSI_PROMPT	00000450-R	1135 (1)	1140 (1)
BELL	=00000007	173 (1)	#-1056 (1) #-557 (1)
BIT...	=00000004	163 (1)	160 (1) 161 (1) 162 (1) 163 (1)
BS	=00000008	174 (1)	#-1044 (1)
BSLASH	=0000005C	188 (1)	#-953 (1) #-963 (1)
CLI$K_BUFSIZ	=00000100	157 (1)	157 (1)
CLI$K_SIZE	00000444	157 (1)	
CLI$L_ADDRESS	00000018	157 (1)	
CLI$L_COMMAND	00000004	157 (1)	
CLI$L_DATA	0000001C	157 (1)	#-415 (1)
CLI$L_FLAGS	00000000	157 (1)	
CLI$L_LAST	00000024	157 (1)	
CLI$L_NEXT	00000030	157 (1)	
CLI$L_PASS	0000002C	157 (1)	
CLI$L_SUBT	00000028	157 (1)	
CLI$L_TEST	00000020	157 (1)	
CLI$Q_BUFQWD	00000034	157 (1)	
CLI$Q_FILE	00000008	157 (1)	
CLI$Q_SECTION	00000010	157 (1)	
CLI$Q_TIME	0000043C	157 (1)	
CLI$T_BUFFER	0000003C	157 (1)	
CLI$V_ADAPTER	=00000018	157 (1)	
CLI$V_ADR	=0000000B	157 (1)	
CLI$V_ASCII	=00000013	157 (1)	
CLI$V_BREAK	=0000000A	157 (1)	
CLI$V_BRIEF	=0000001B	157 (1)	
CLI$V_BYTE	=0000000D	157 (1)	
CLI$V_CLEAR	=00000002	157 (1)	
CLI$V_DEC	=00000010	157 (1)	
CLI$V_DEFAULT	=0000000C	157 (1)	
CLI$V_DEPOSIT	=00000019	157 (1)	
CLI$V_EVENT	=00000008	157 (1)	
CLI$V_EXAM	=00000005	157 (1)	
CLI$V_FLAGS	=00000009	157 (1)	
CLI$V_HEX	=00000012	157 (1)	
CLI$V_KERNEL	=00000017	157 (1)	
CLI$V_LOAD	=00000006	157 (1)	
CLI$V_LONG	=0000000F	157 (1)	
CLI$V_NOTNUF	=00000001	157 (1)	
CLI$V_OCT	=00000011	157 (1)	
CLI$V_PREG	=0000001A	157 (1)	
CLI$V_QA	=00000007	157 (1)	
CLI$V_QACKLOOPLOOPS	=0000001C	157 (1)	
CLI$V_QAERRORPRINTS	=0000001B	157 (1)	
CLI$V_QAMULTIPLEPASS	=0000001F	157 (1)	
CLI$V_QASUBTESTLOOPS	=0000001E	157 (1)	
CLI$V_QATESTLOOPS	=0000001D	157 (1)	

```
CLI$V_REG                       =00000014      157     (1)
CLI$V_REQUIRED                  =00000000      157     (1)
CLI$V_RUN                       =00000015      157     (1)
CLI$V_SET                       =00000003      157     (1)
CLI$V_SHOW                      =00000004      157     (1)
CLI$V_VALSEC                    =00000016      157     (1)
CLI$V_WORD                      =0000000E      157     (1)
CMK$RCONSOLE                     00000109-R    615     (1)   #-1175  (1)   #-540   (1)   #-606   (1)
CMK$TCONSOLE                     000002B2-R    823     (1)   #-757   (1)   #-817   (1)
CMK$_                           =00000004      160     (1)
CMK$_ASTEXIT                    =00000000      160     (1)
CMK$_CANTIM                     =0000000B      160     (1)
CMK$_HIBER                      =00000007      160     (1)
CMK$_INITSCB                    =00000008      160     (1)
CMK$_LAST                       =0000000E      160     (1)
CMK$_MMENABLE                   =00000003      160     (1)
CMK$_RCONSOLE                   =00000001      160     (1)   #-1175  (1)   #-540   (1)   #-606   (1)
CMK$_REFRESH                    =00000005      160     (1)
CMK$_SCHDWK                     =00000006      160     (1)
CMK$_SETIMR                     =0000000C      160     (1)
CMK$_SETIPL                     =00000009      160     (1)
CMK$_SETPRT                     =0000000D      160     (1)
CMK$_SGIPR                      =0000000A      160     (1)
CMK$_TCONSOLE                   =00000002      160     (1)   #-757   (1)   #-817   (1)
CNTRLC                          =00000003      172     (1)   #-548   (1)   #-630   (1)   #-972   (1)
CNTRLO                          =0000000F      180     (1)   #-551   (1)   #-663   (1)
CNTRLQ                          =00000011      181     (1)   #-544   (1)   #-658   (1)
CNTRLR                          =00000012      182     (1)   #-979   (1)
CNTRLS                          =00000013      183     (1)   #-554   (1)   #-653   (1)
CNTRLU                          =00000015      184     (1)   #-966   (1)
COLUMN                           00000000-R    197     (1)   #-759   (1)   #-793   (1)   #-795   (1)   #-808   (1)
                                                             #-814   (1)   #-928   (1)
CONSFLG                          00000003-R    203     (1)   #-632   (1)   #-655   (1)   #-660   (1)   #-665   (1)
                                                             #-787   (1)
CR                              =0000000D      179     (1)   #-1001  (1)   #-1009  (1)     211   (1)     214   (1)
                                                               220   (1)     223   (1)   #-791   (1)
CRD$M_CRD_DEBUG                 =00000001      163     (1)
CRD$M_CTRL_C_NO_ECHO            =00000002      163     (1)
CRD$M_FLUSH_CTRL_C              =00000004      163     (1)
CRD$M_INHIBIT_ALLOCATE          =00000008      163     (1)
CRD$V_CRD_DEBUG                 =00000000      163     (1)
CRD$V_CTRL_C_NO_ECHO            =00000001      163     (1)   #-634   (1)
CRD$V_FLUSH_CTRL_C              =00000002      163     (1)   #-647   (1)   #-648   (1)   #-652   (1)
CRD$V_INHIBIT_ALLOCATE          =00000003      163     (1)
DEC_PROMPT                       0000044D-R    1134    (1)     1139  (1)
DELETE                          =0000007F      189     (1)   #-945   (1)
DS$GB_WIDTH                      00000002-R    201     (1)   #-419   (1)   #-430   (1)   #-485   (1)   #-795   (1)
DS$GL_CRD_FLAGS                  00000000-XR                    634   (1)     647   (1)     648   (1)     652   (1)
DS$GL_FLAGS                      00000000-XR                    603   (1)     639   (1)     640   (1)     649   (1)
                                                             #-668   (1)     695   (1)     780   (1)     873   (1)
                                                             #-925   (1)     943   (1)     952   (1)     962   (1)
DS$GL_TERMCHAR                   00000000-XR                #-1156   (1)    1195   (1)   #-1196   (1)    1214   (1)
                                                             #-1215  (1)
DS$GW_TTOUT                      00000000-XR                #-428   (1)   #-435   (1)
DS$K_PRINTB                     =00000002      162     (1)
DS$K_PRINTF                     =00000001      162     (1)   #-442   (1)   #-490   (1)
DS$K_PRINTI                     =00000000      162     (1)
```

```
DS$K_PRINTX                     =00000003     162     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     162     (1)
DS$K_TYPE_ABORT_TEST            =00000013     162     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     162     (1)     #-442     (1)
DS$K_TYPE_COMMAND_OUT           =00000016     162     (1)     #-490     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     162     (1)
DS$K_TYPE_DS_PROMPT             =00000001     162     (1)
DS$K_TYPE_DS_START              =0000001D     162     (1)
DS$K_TYPE_ERRDEV                =00000008     162     (1)
DS$K_TYPE_ERRHARD               =00000006     162     (1)
DS$K_TYPE_ERROR_BODY            =00000009     162     (1)
DS$K_TYPE_ERROR_END             =0000000A     162     (1)
DS$K_TYPE_ERRPREP               =0000001B     162     (1)
DS$K_TYPE_ERRSOFT               =00000007     162     (1)
DS$K_TYPE_ERRSUP                =00000C04     162     (1)
DS$K_TYPE_ERRSYS                =00000005     162     (1)
DS$K_TYPE_ERR_HALT              =0000000D     162     (1)
DS$K_TYPE_EXCEPTION             =0000000C     162     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     162     (1)
DS$K_TYPE_FIRST_PASS            =00000011     162     (1)
DS$K_TYPE_GENERAL               =00000000     162     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     162     (1)
DS$K_TYPE_NO_TESTS              =00000012     162     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     162     (1)
DS$K_TYPE_PROGRAM_END           =00000010     162     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     162     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     162     (1)
DS$K_TYPE_QIO_INVADP            =00000024     162     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     162     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     162     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     162     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     162     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     162     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     162     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     162     (1)
DS$K_TYPE_START_ERR             =00000018     162     (1)
DS$K_TYPE_START_LIST            =00000025     162     (1)
DS$K_TYPE_SUMMARY               =0000000E     162     (1)
DS$K_TYPE_USER_PROMPT           =00000002     162     (1)
DS$M_ABRTFLG                    =00000040     161     (1)
DS$M_BADTIME                    =00100000     161     (1)
DS$M_BATCH                      =00400000     161     (1)
DS$M_BRKCLR                     =00001000     161     (1)
DS$M_BRKPT                      =00000800     161     (1)
DS$M_CHARFLG                    =00000100     161     (1)
DS$M_CMDFLG                     =00000080     161     (1)
DS$M_CTRLC                      =00000001     161     (1)
DS$M_CTRLO                      =00010000     161     (1)     #-668     (1)     #-924     (1)
DS$M_DEVFLG                     =00000200     161     (1)
DS$M_DISABLCC                   =01000000     161     (1)
DS$M_DONFLG                     =00002000     161     (1)
DS$M_ERRFLG                     =00000010     161     (1)
DS$M_EXCEPT                     =00080000     161     (1)
DS$M_EXETST                     =00040000     161     (1)
DS$M_HLTFLG                     =00000008     161     (1)
DS$M_LODFLG                     =00000002     161     (1)
DS$M_MEMMGT                     =00008000     161     (1)
```

```
DS$M_OUTPUT                     =00800000      161     (1)     #-924     (1)
DS$M_RUBFLG                     =00000020      161     (1)
DS$M_SCRIPT                     =00200000      161     (1)
DS$M_SETIMR                     =02000000      161     (1)
DS$M_STRFLG                     =00000004      161     (1)
DS$M_SUBT                       =00004000      161     (1)
DS$M_SYSFLG                     =00000400      161     (1)
DS$M_TIMRON                     =00020000      161     (1)
DS$V_ABRTFLG                    =00000006      161     (1)
DS$V_BADTIME                    =00000014      161     (1)
DS$V_BATCH                      =00000016      161     (1)
DS$V_BRKCLR                     =0000000C      161     (1)
DS$V_BRKPT                      =0000000B      161     (1)
DS$V_CHARFLG                    =00000008      161     (1)
DS$V_CMDFLG                     =00000007      161     (1)
DS$V_CTRLC                      =00000000      161     (1)     #-640     (1)     #-649     (1)
DS$V_CTRLO                      =00000010      161     (1)     #-639     (1)     #-780     (1)     #-873     (1)     #-943     (1)
DS$V_DEVFLG                     =00000009      161     (1)
DS$V_DISABLCC                   =00000018      161     (1)
DS$V_DONFLG                     =0000000D      161     (1)
DS$V_ERRFLG                     =00000004      161     (1)
DS$V_EXCEPT                     =00000013      161     (1)
DS$V_EXETST                     =00000012      161     (1)
DS$V_HLTFLG                     =00000003      161     (1)
DS$V_LODFLG                     =00000001      161     (1)
DS$V_MEMMGT                     =0000000F      161     (1)
DS$V_OUTPUT                     =00000017      161     (1)     #-602     (1)     #-702     (1)     #-709     (1)
DS$V_RUBFLG                     =00000005      161     (1)     #-951     (1)     #-961     (1)
DS$V_SCRIPT                     =00000015      161     (1)
DS$V_SETIMR                     =00000019      161     (1)
DS$V_STRFLG                     =00000002      161     (1)
DS$V_SUBT                       =0000000E      161     (1)
DS$V_SYSFLG                     =0000000A      161     (1)
DS$V_TIMRON                     =00000011      161     (1)
DSA$GL_FLAGS                     0000FE00                       421     (1)       783     (1)       785     (1)
DSA$V_APT                       =0000001F                   #-783     (1)
DSA$V_NORPT                     =0000001B                   #-784     (1)
DSA$V_USER                      =0000001C                   #-421     (1)
DSX$PRINT                        00000000-XR                  442     (1)       490     (1)
END_HARDCOPY_TYPES               00000138-R     360     (1)
END_VIDEO_TYPES                  0000010B-R     336     (1)
EVENT_FLAGS                      00000004-R     205     (1)      1172     (1)
FF                              =0000000C       178     (1)   #-1040     (1)
HARDCOPY_TYPES                   0000010F-R     343     (1)      1191     (1)
HYPHEN                          =0000002D       186     (1)
IO$V_CANCTRLO                    00000000-XR                  #-871     (1)
IO$_READPROMPT                   00000000-XR                  #-926     (1)
IO$_SENSEMODE                    00000000-XR                  #-428     (1)
IO$_SETMODE                      00000000-XR                  #-435     (1)
KB_POLL                          000000AC-R     538     (1)   #-779     (1)   #-788     (1)
KB_POLL_X                        000000DD-R     560     (1)   #-542     (1)   #-545     (1)     #-549     (1)     #-552     (1)
                                                              #-555     (1)
LA36_PROMPT                      00000454-R    1136     (1)      1141     (1)
LASTCHAR                         00000001-R     199     (1)   #-816     (1)   #-930     (1)
LF                              =0000000A       176     (1)   #-1006     (1)   #-1036     (1)      211     (1)      214     (1)
                                                               220     (1)      223     (1)   #-930     (1)
LOWCASE                         =00000020       171     (1)
```

```
MAX_WAIT                        0000047A-R  1150  (1)      1155  (1)
OUT_ASCIC                       0000021A-R   763  (1)  #-698   (1)  #-799   (1)  #-934   (1)  #-969   (1)
                                                      #-986   (1)
OUT_CHAR                        0000022F-R   778  (1)  #-1005  (1)  #-1007  (1)  #-1048  (1)  #-1057  (1)
                                                      #-1061  (1)  #-703   (1)  #-710   (1)  #-772   (1)
                                                      #-954   (1)  #-957   (1)  #-964   (1)
OUT_CHARX                       000002B1-R   819  (1)  #-780   (1)  #-785   (1)
OUT_CNTRL                       000001F6-R   752  (1)  #-1168  (1)  #-637   (1)  #-667   (1)
OUT_ECHO                        0000023A-R   782  (1)  #-558   (1)
OUT_STRING                      0000021D-R   766  (1)  #-876   (1)  #-937   (1)
PARITY                          =00000080    170  (1)  #-944   (1)
PR$_RXCS                        =00000020             #-618   (1)
PR$_RXDB                        =00000021             #-622   (1)
PR$_TXCS                        =00000022             #-826   (1)  #-827   (1)
PR$_TXDB                        =00000023             #-829   (1)
PREADVBLK                       000002E4-R   922  (1)  #-970   (1)
PROMPTS                         00000456-R  1138  (1)      1157  (1)
PSL$V_IS                        =0000001A    160  (1)  #-1175  (1)  #-540   (1)  #-606   (1)  #-757   (1)
                                                      #-817   (1)
QIO$_ASTADR                     =00000014    159  (1)
QIO$_ASTPRM                     =00000018    159  (1)
QIO$_CHAN                       =00000008    159  (1)
QIO$_EFN                        =00000004    159  (1)
QIO$_FUNC                       =0000000C    159  (1)      872  (1)  #-926   (1)
QIO$_IOSB                       =00000010    159  (1)     1066  (1)
QIO$_NARGS                      =0000000C    159  (1)
QIO$_P1                         =0000001C    159  (1)      705  (1)  #-875   (1)      939  (1)
QIO$_P2                         =00000020    159  (1)  #-1053  (1)
QIO$_P3                         =00000024    159  (1)
QIO$_P4                         =00000028    159  (1)  #-1013  (1)
QIO$_P5                         =0000002C    159  (1)  #-699   (1)  #-936   (1)
QIO$_P6                         =00000030    159  (1)
QUAD_TIME                       00000489-R  1151  (1)      1155  (1)      1161  (1)
QUESTN                          =0000003F    187  (1)
READVBLK                        000002E4-R   923  (1)
RECVD_BUF                       00000466-R  1149  (1)  #-1169  (1)      1170  (1)  #-1179  (1)      1208  (1)
RECVD_BUF_LENGTH                =00000014   1144  (1)     1149  (1)
RINPUT                          000000E0-R   601  (1)  #-607   (1)  #-609   (1)  #-942   (1)
RINPUTX                         00000108-R   612  (1)
RTYPE                           000001BE-R   694  (1)  #-604   (1)  #-987   (1)
SIZ...                          =00000001    163  (1)      161  (1)      163  (1)
SIZE_TERM                       00000491-R  1153  (1)
SPACE                           =00000020    185  (1)  #-805   (1)
SS$_CONTROLC                    00000000-XR           #-976   (1)
SS$_WASSET                      00000000-XR           #-1173  (1)
SYS$BINTIM                      00000000-XR              1155  (1)
SYS$QIOW                        00000000-XR               428  (1)      435  (1)
SYS$READEF                      00000000-XR              1172  (1)
SYS$SETEF                       00000000-XR              1071  (1)      877  (1)
SYS$SETIMR                      00000000-XR              1161  (1)
TAB                             =00000009    175  (1)  #-1042  (1)  #-803   (1)
TT$M_SCOPE                      =00001000             #-1186  (1)
TT$_LA100                       =00000025                350  (1)
TT$_LA12                        =00000024                347  (1)
TT$_LA120                       =00000021                353  (1)
TT$_LA34                        =00000022                356  (1)      359  (1)
TT$_LA36                        =00000020             #-1194  (1)      364  (1)
```

```
TT$_UNKNOWN                     =00000000                                299     (1)     302     (1)
TT$_VT100                       =00000060                                266     (1)     275     (1)     278     (1)     281     (1)
                                                                         284     (1)     287     (1)     290     (1)     293     (1)
                                                                         296     (1)
TT$_VT52                        =00000040                                257     (1)     260     (1)     263     (1)
TT$_VT55                        =00000041                                248     (1)
TT$_VT5X                        =00000040                                239     (1)     242     (1)     245     (1)     251     (1)
                                                                         254     (1)
T_CNTRLC                         00000000-R     210     (1)              636     (1)
T_CNTRLO                         0000000F-R     219     (1)              666     (1)
T_CNTRLR                         0000000C-R     216     (1)              985     (1)
T_CNTRLU                         00000007-R     213     (1)              968     (1)
T_CRLF                           00000014-R     222     (1)              697     (1)     798     (1)     933     (1)
T_LA100                          00000117-R     348     (1)
T_LA12                           0000010F-R     345     (1)
T_LA120                          0000011F-R     351     (1)
T_LA34                           00000126-R     354     (1)
T_LA34A                          0000012F-R     357     (1)
T_LA36                           0000013C-R     363     (1)
T_SHOW_WIDTH                     00000048-R     228     (1)              490     (1)
T_VT100                          0000009F-R     273     (1)
T_VT100A                         0000009A-R     264     (1)
T_VT100B                         000000A8-R     276     (1)
T_VT100C                         000000B1-R     279     (1)
T_VT100D                         000000BA-R     282     (1)
T_VT100E                         000000C3-R     285     (1)
T_VT100F                         000000CC-R     288     (1)
T_VT100G                         000000D5-R     291     (1)
T_VT100H                         000000DE-R     294     (1)
T_VT220                          000000E7-R     297     (1)
T_VT240                          000000F7-R     300     (1)
T_VT50_A                         0000006D-R     237     (1)
T_VT50_B                         00000072-R     240     (1)
T_VT50_C                         00000077-R     243     (1)
T_VT50_H                         00000081-R     249     (1)
T_VT50_J                         00000086-R     252     (1)
T_VT52_K                         0000008B-R     255     (1)
T_VT52_L                         00000090-R     258     (1)
T_VT52_M                         00000095-R     261     (1)
T_VT55_E                         0000007C-R     246     (1)
T_WIDTH                          00000017-R     225     (1)               442     (1)
VIDEO_TYPES                      0000006D-R     235     (1)              1187     (1)
VRSETWIDTH                       00000000-R     414     (1)
VRSHOWWIDTH                      0000008C-R     484     (1)
VT                              =0000000B       177     (1)             #-1038   (1)
WRITEVBLK                        000002C8-R     870     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ASNPUSH	1	1161 (1)	1161 (1)
$BINTIM_S	1	1155 (1)	1155 (1)
$D1_PRINT_S	1	442 (1)	442 (1) 490 (1)
$DEF	1	163 (1)	
$DEFINI	1	156 (1)	156 (1) 158 (1) 164 (1)
$DS_DSADEF	5	156 (1)	156 (1)
$DS_TYPEDEF	4	162 (1)	162 (1)
$EQU	1	163 (1)	160 (1) 162 (1)
$EQULS1	1	162 (1)	160 (1) 162 (1)
$EQULST	1	160 (1)	160 (1) 162 (1)
$GBLINI	2		160 (1) 161 (1) 162 (1) 163 (1)
$OFFDEF	1	159 (1)	159 (1)
$PRDEF	4	158 (1)	158 (1)
$PRINT	2	440 (1)	440 (1) 487 (1)
$PUSHADR	1	428 (1)	1155 (1) 1161 (1) 1172 (1) 428 (1) 435 (1)
$PUSHTWO	1	428 (1)	428 (1) 435 (1)
$QIODEF	1	159 (1)	159 (1)
$QIOPUSH	1	428 (1)	1161 (1) 428 (1) 435 (1)
$QIOW_S	1	425 (1)	425 (1) 432 (1)
$READEF_S	1	1172 (1)	1172 (1)
$SETEF_G	1	877 (1)	1071 (1) 877 (1)
$SETIMR_S	1	1161 (1)	1161 (1)
$TTDEF	6	164 (1)	164 (1)
$VIELD	1		161 (1) 163 (1)
$VIELD1	1	163 (1)	161 (1) 163 (1)
BR_IF_CTRLO	1	780 (1)	780 (1)
BR_IF_CTRL_C_NO_ECHO	1	634 (1)	634 (1)
BR_IF_FLUSH_CTRL_C	1	647 (1)	647 (1)
BR_IF_NOT_APT	1	783 (1)	783 (1)
BR_IF_NOT_USER	1	421 (1)	421 (1)
CLEAR_CTRLO	1	873 (1)	873 (1) 943 (1)
CLEAR_FLUSH_CTRL_C	1	652 (1)	652 (1)
CLIDEF	3	157 (1)	157 (1)
CMK	1	540 (1)	1175 (1) 540 (1) 606 (1) 757 (1) 817 (1)
CMKDEF	1	160 (1)	160 (1)
CRDDEF	1	163 (1)	163 (1)
DSFDEF	3	161 (1)	161 (1)
SET_CTRLC	1	640 (1)	640 (1) 649 (1)
SET_CTRLO	1	639 (1)	639 (1)
SET_FLUSH_CTRL_C	1	648 (1)	648 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.12	00:00:00.27
Command processing	147	00:00:00.72	00:00:01.69
Pass 1	887	00:00:12.46	00:00:28.73
Symbol table sort	0	00:00:00.96	00:00:02.42
Pass 2	307	00:00:03.37	00:00:05.34
Symbol table output	40	00:00:00.26	00:00:00.42
Psect synopsis output	8	00:00:00.04	00:00:00.27
Cross-reference output	146	00:00:01.38	00:00:02.04
Assembler run totals	1572	00:00:19.33	00:00:41.20

The working set limit was 1000 pages.
66786 bytes (131 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 572 non-local and 79 local symbols.
1219 source lines were read in Pass 1, producing 0 object records in Pass 2.
83 pages of virtual memory were used to define 41 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	3
DRB1:[DS.WORK]DS.MLB;218	16
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	37

702 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CONSOLE/UPDA=(CONSOLE.UPD,CONSOLE.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SY
```

```
;  0001  0     %TITLE '*** Loading and Unloading the CRD Image'
;  0002  0     MODULE CRDIMAGE (                   ! Routines to handle loading and unloading the Loadable-CRD image
;  0003  0                     IDENT = '02-11'
;  0004  0                     ) =
;  0005  0     !++
;  0006  0     ! COPYRIGHT (c) 1980, 1981
;  0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;  0008  0     !
;  0009  0     ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
;  0010  0     ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
;  0011  0     ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;  0012  0     ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;  0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;  0014  0     ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
;  0015  0     ! REMAIN IN DEC.
;  0016  0     !
;  0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;  0018  0     ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
;  0019  0     ! CORPORATION.
;  0020  0     !
;  0021  0     ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
;  0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;  0023  0     !--
;  0024  0     !++
;  0025  0     ! Facility:
;  0026  0     !
;  0027  0     !       Diagnostic Supervisor (part of the Resident-DS image).
;  0028  0     !
;  0029  0     ! Abstract:
;  0030  0     !
;  0031  0     !       This module contains the routines necessary to load the CRD-Loadable image file and integrate
;  0032  0     !       it with the Resident-DS, thus putting the Supervisor under control of CRD. It also now contains
;  0033  0     !       the routines necessary to unload the CRD image.
;  0034  0     !
;  0035  0     ! Author:
;  0036  0     !
;  0037  0     !       Jack Stansbury
;  0038  0     !
;  0039  0     ! Creation Date:
;  0040  0     !
;  0041  0     !       29-March-1982
;  0042  0     !
;  0043  0     ! Version
;  0044  0     !
;  0045  0     !       Version 6.9
;  0046  0     !
;  0047  0     ! Modified By:
;  0048  0     !
;  0049  0     !       01      Jack Stansbury, 29-June-1982, Version 6.8
;  0050  0     !               Took out checking the DS verification bytes, and the check to see if
;  0051  0     !               the DS vector count EQL'ed the CRD vector count (why was this in there
;  0052  0     !               anyway???). Assume that the DS verification bytes are okay. It is CRD
;  0053  0     !               that I am verifying anyway!
;  0054  0     !
;  0055  0     !       02      Jack Stansbury, 1-July-1982, Version 6.8
;  0056  0     !               You dummy! I put the check of the vector count in there for a good reason!
;  0057  0     !               If the vector counts are not the same, one or the other may get some other
```

```
;    0058  0     !                  data overwritten (i.e., data following the dispatch vectors). This must
;    0059  0     !                  NOT happen. So, if the vector counts are not equal, it is an incompat-
;    0060  0     !                  ibility error (not a corrupted error).
;    0061  0     !
;    0062  0     !          03      Jack Stansbury, 15-July-1982, Version 6.9
;    0063  0     !                  Added support for loading in the CRD Menu Test file. Which file to load in
;    0064  0     !                  depends on which DSA bit is set.
;    0065  0     !
;    0066  0     !          04      Jack Stansbury, 25-July-1982, Version 6.9
;    0067  0     !                  Added code to check to see if this was running in User-Mode, and if so, then
;    0068  0     !                  use the CRD$ExpReg routine; else use $ExpReg to expand the VDS's memory.
;    0069  0     !                  Note that although CRD Menu Test will NOT be normally run in User Mode,
;    0070  0     !                  it is run this way when it is being debugged. Whether or not it should run
;    0071  0     !                  is determined by the CLI routines.
;    0072  0     !
;    0073  0     !          05      Jack Stansbury, 7-September-1982, Version 6.9
;    0074  0     !                  Made major modifications to all the routines in this module. I renamed it from
;    0075  0     !                  LoadCRD to CRDImage. Added many routines.
;    0076  0     !
;    0077  0     !          06      Jack Stansbury, 26-Oct-1982, Version 6.9
;    0078  0     !                  Added code to disable printing ^C's during running of CRD.
;    0079  0     !
;    0080  0     !          07      Jack Stansbury, -2-Mar-1983, Version 6.11
;    0081  0     !                  Improved the debugging mode of CRD. Added SET CRD/DEBUG to the CLI.
;    0082  0     !                  Made printout of debug vectors much better and much more informative.
;    0083  0     !                  Put the actual debug vector printout in this module rather than in a
;    0084  0     !                  CRD module. Also took out references to the DS$K_Numb_Dispatch_Vectors.
;    0085  0     !                  Also added the DSR$Print_Typecode_Name routine, for use by all the VDS.
;    0086  0     !                  Also changed the check to see if the debug flag was set (in CRD_Error).
;    0087  0     !
;    0088  0     !          08      Jack Stansbury, 7-Mar-1983, Version 6.11
;    0089  0     !                  Added the DSR$Show_Calls routine - this is a global routine (JSB linkage)
;    0090  0     !                  that will print the contents of the last 'n' call frames on the stack.
;    0091  0     !
;    0092  0     !          09      Jack Stansbury, 8-Mar-1983, Version 6.11
;    0093  0     !                  Took out the DSR$Show_Calls routine. I created a new module for this routine
;    0094  0     !                  called CallFrame.B32. That routine didn't realy belong in here.
;    0095  0     !
;    0096  0     !          10      John Ciukaj     4-April-1983    Version 6.11
;    0097  0     !                  - Changed references to CRD Bits in DSA Flags
;    0098  0     !                    to distinguish between online and offline
;    0099  0     !
;    0100  0     !          11      Jack Stansbury, April 4, 1983, Version 6.11
;    0101  0     !                  Made Exchange_Dispatch_Vectors into a global routine so it can
;    0102  0     !                  be accessed through the debugger (DEBUG-32).
;    0103  0     !--
;    0104  0     %SBTTL  'Libraries'
;    0105  1     BEGIN                                   ! Begin the CRDIMAGE module
;    0106  1
;    0107  1     LIBRARY
;    0108  1             '$DS';
;    0109  1
;    0110  1     LIBRARY
;    0111  1             '$Diag';
;    0112  1
;    0113  1     LIBRARY
;    0114  1             '$CRD';
```

```
;    0115  1
;    0116  1     LIBRARY
;    0117  1             'Sys$Library:Lib';
;    0118  1     %SBTTL  'Table of Contents'
;    0119  1
;    0120  1     FORWARD ROUTINE
;    0121  1             Exchange_Dispatch_Vectors       : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0122  1                                              ! Exchange the dispatch vectors between the DS and CRD
;    0123  1
;    0124  1             DSR$Load_CRD                    : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0125  1                                              ! Load the CRD Loadable image file
;    0126  1
;    0127  1             DSR$Unload_CRD                  : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0128  1                                              ! Unload the CRD Loadable image file
;    0129  1
;    0130  1             DSR$Restore_CRD_Vectors         : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0131  1                                              ! Restore the CRD dispatch vectors - this is called by the BEGIN0 label
;    0132  1                                              ! in the KERNEL module. The Unload_CRD routine cannot be called because
;    0133  1                                              ! memory has already been re-mapped by the time BEGIN0 executes.
;    0134  1
;    0135  1             CRD_Exit                        : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0136  1                                              ! Either exit from the VDS image (back to the console), or to the CLI.
;    0137  1
;    0138  1             CRD_Error                       : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0139  1                                              ! Print an error message and take appropriate action(s).
;    0140  1
;    0141  1             Internal_Error                  : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),                    !E07
;    0142  1                                              ! Print the extended debugging information about CRD.         !E07
;    0143  1
;    0144  1             DSR$Print_TypeCode_Name         : NOVALUE ADDRESSING_MODE (LONG_RELATIVE);                    !E07
;    0145  1                                              ! Print the message                                           !E07
;    0146  1     %SBTTL  'Included Macros and Literals'
;    0147  1
;    0148  1             $CRD_Literals;                  ! Define the CRD literals
;    0149  1             $DS_DSSDef;                     ! Define the DS service routine vector addresses
;    0150  1             $DS_TypeDef;                    ! Define the typecode literals
;    0151  1             $DS_DSADef;                     ! Define the DSA flags.
;    0152  1     %SBTTL  'Other Literals'
;    0153  1
;    0154  1     LITERAL
;    0155  1             !+
;    0156  1             ! The SID Type field for the VAX'en (see the SRM for more on this)
;    0157  1             !-
;    0158  1
;    0159  1             SID_780 = 1,
;    0160  1             SID_750 = 2,
;    0161  1             SID_730 = 3;
;    0162  1
;    0163  1     LITERAL
;    0164  1             !+
;    0165  1             ! These indicate to the Error routine whether or not the space has been allocated
;    0166  1             ! for the CRD image.
;    0167  1             !-
;    0168  1
; P  0169  1             $EquLst (K_, LOCAL, 0, 1
; P  0170  1                     ,(Region_Expanded)
; P  0171  1                     ,(Region_Not_Expanded)
```

```
;    0172  1                     ),
;    0173  1
;    0174  1             !+
;    0175  1             ! These are the types of errors that are detected and reported.
;    0176  1             !-
;    0177  1
; P  0178  1             $EquLst (K_, LOCAL, 0, 1
; P  0179  1                     ,(No_DSA_Bit_Set)
; P  0180  1                     ,(Wrong_CPU)
; P  0181  1                     ,(Determine_Size)
; P  0182  1                     ,(Expand_Region)
; P  0183  1                     ,(Load_File)
; P  0184  1                     ,(CRD_Internal)
; P  0185  1                     ,(Corrupted)
; P  0186  1                     ,(Incompatible)
; P  0187  1                     ,(Not_Contracted)
; P  0188  1                     ,(MM_Is_On)
; P  0189  1                     ,(Last_Error_Type)
;    0190  1                     ),
;    0191  1
;    0192  1             !+
;    0193  1             ! These are used to reference the first longword in both the DS$AL_DS_Dispatch_Vectors
;    0194  1             ! and the CRD$AL_CRD_Dispatch_Vectors longwords.
;    0195  1             !-
;    0196  1
; P  0197  1             $EquLst (K_, LOCAL, 0, 1
; P  0198  1                     ,(Number_Of_Vectors)
; P  0199  1                     ,(Positive_Version)
; P  0200  1                     ,(Negative_Version)
; P  0201  1                     ,(Null_Byte)
;    0202  1                     ),
;    0203  1
;    0204  1             !+
;    0205  1             ! Use these to reference the CRD_Image_Location array.
;    0206  1             !-
;    0207  1
; P  0208  1             $EquLst (K_, LOCAL, 0, 1
; P  0209  1                     ,(Starting_Adr)
; P  0210  1                     ,(Ending_Adr)
;    0211  1                     ),
;    0212  1
;    0213  1             !+
;    0214  1             ! Use these to reference the DS$Q_CRD_Image_Desc descriptor. Note that this is not a normal
;    0215  1             ! descriptor, since the size field is a longword, rather than a word. This is so because the
;    0216  1             ! size of the CRD image may be bigger than 65536 (or whatever that number - 2^16 - is).
;    0217  1             !-
;    0218  1
; P  0219  1             $EquLst (K_, LOCAL, 0, 1
; P  0220  1                     ,(Image_Size)
; P  0221  1                     ,(Image_Ptr)
;    0222  1                     );
;    0223  1     %SBTTL  'External Routines'
;    0224  1
;    0225  1     EXTERNAL ROUTINE
;    0226  1             DS_Cleanup                      : JSB_None ADDRESSING_MODE (LONG_RELATIVE),
;    0227  1                                             ! Cleanup the diagnostic before going back to CLI command mode
;    0228  1
```

```
;    0229  1              MapFree                        : ADDRESSING_MODE (LONG_RELATIVE),
;    0230  1                                             ! Map all of free memory
;    0231  1
;    0232  1              CRD$ExpReg                     : ADDRESSING_MODE (LONG_RELATIVE),
;    0233  1                                             ! Expand Region for CRD.
;    0234  1
;    0235  1              DSR$Check_MenuTest_Off_Set     : JSB_None ADDRESSING_MODE (LONG_RELATIVE),
;    0236  1                                             !                                          [10]
;    0237  1                                             ! Return 1 if the MenuTest bits are set in the R5 passed to the VDS
;    0238  1
;    0239  1              DSR$Check_AutoTest_Off_Set     : JSB_None ADDRESSING_MODE (LONG_RELATIVE),
;    0240  1                                             !                                          [10]
;    0241  1                                             ! Return 1 if the AutoTest bits are set in the R5 passed to the VDS
;    0242  1
;    0243  1              Exe$AloNonPaged                : JSB_Alloc ADDRESSING_MODE (LONG_RELATIVE),
;    0244  1                                             ! Allocate a block of memory from non-paged pool
;    0245  1
;    0246  1              Exe$DeaNonPaged                : JSB_Deall ADDRESSING_MODE (LONG_RELATIVE),
;    0247  1                                             ! Deallocate a block of memory from non-paged pool
;    0248  1
;    0249  1              DSV$Exit                       : JSB_None ADDRESSING_MODE (LONG_RELATIVE);
;    0250  1                                             ! Use a JSB instruction to get to this routine. Use this routine to
;    0251  1                                             ! cause the Resident-DS to exit.
;    0252  1      %SBTTL  'External Own Storage'
;    0253  1
;    0254  1      EXTERNAL
;    0255  1              Begin_Bliss                    : ADDRESSING_MODE (LONG_RELATIVE),
;    0256  1                                             ! The location of the Begin_Bliss label (in the KERNEL module).
;    0257  1
;    0258  1              DS$GA_DS_Ctrl_C_First          : ADDRESSING_MODE (LONG_RELATIVE),
;    0259  1                                             ! The first ^C handler for the VDS
;    0260  1
;    0261  1              DS$GA_DS_Ctrl_C_Second         : ADDRESSING_MODE (LONG_RELATIVE),
;    0262  1                                             ! The second ^C handler for the VDS
;    0263  1
;    0264  1              DS$L_UserCntrlC                : ADDRESSING_MODE (LONG_RELATIVE),
;    0265  1                                             ! The user's ^C handler
;    0266  1
;    0267  1              DS$GL_Flags                    : ADDRESSING_MODE (LONG_RELATIVE),
;    0268  1                                             ! The DS flags longword
;    0269  1
;    0270  1              DS$AL_DS_Dispatch_Vectors      : ADDRESSING_MODE (LONG_RELATIVE),
;    0271  1                                             ! The start of the Resident-DS dispatch vectors
;    0272  1
;    0273  1              DS$GL_CRD_Flags                : ADDRESSING_MODE (LONG_RELATIVE),
;    0274  1                                             ! The DS/CRD flags longword in the DSVECS module
;    0275  1
;    0276  1              DS$GA_LastAdr                  : ADDRESSING_MODE (LONG_RELATIVE),
;    0277  1                                             ! The last address at the top of the Supervisor image (after the memory
;    0278  1                                             ! has been allocated for the QIO database and the SCRIPT junk).
;    0279  1
;    0280  1              DS$GL_CRD_Debug                : ADDRESSING_MODE (LONG_RELATIVE);                                  !C07
;    0281  1                                             ! This longword contains either 0 or 1. 1 implies CRD will print CRD !C07
;    0282  1                                             ! debugging messages. 0 implies it will not.                        !C07
;    0283  1      %SBTTL  'Psect Definitions'
;    0284  1
;    0285  1      PSECT
```

```
;    0286  1        PLIT    = Data (NOEXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0287  1
;    0288  1    PSECT
;    0289  1        CODE    = Code (  EXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0290  1
;    0291  1    PSECT
;    0292  1        OWN     = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0293  1
;    0294  1    PSECT
;    0295  1        GLOBAL = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0296  1    %SBTTL  'Module-Global Static Storage'
;    0297  1
;    0298  1            ModNam ('CRDIMAGE');              ! Module name for ErrSup macro
;    0299  1    %SBTTL  'Local ASCIC Bindings'
;    0300  1
;    0301  1    BIND
;    0302  1            T_CRD_Image_Name                  = $ASCID ('ENSAB.EXE;0'),                                  !C03
;    0303  1                                              ! The name of the CRD Loadable Image file                  !C03
;    0304  1
;    0305  1            T_CRD_Image_Default               = $ASCID (''),                                             !C03
;    0306  1                                              ! The default name for the CRD Loadable Image file         !C03
;    0307  1
;    0308  1
;    0309  1            T_Null                            = $ASCIC (''),
;    0310  1            T_Menu                            = $ASCIC ('MENU TEST'),                                    !C03
;    0311  1            T_Auto                            = $ASCIC ('AUTO TEST'),                                    !C03
;    0312  1
;    0313  1
;    0314  1        T_Invalid_CPU                         = $ASCIC ('!/CRD !AC ONLY SUPPORTED ON A VAX-11/730,725!/'),
;    0315  1        T_File_Not_Found                      = $ASCIC ('!/UNABLE TO FIND THE LOADABLE PORTION OF CRD !AC!/'),   !C03
;  P 0316  1        T_Unknown_Error                       = $ASCIC ('!/ERROR WHILE ATTEMPTING TO LOAD THE LOADABLE PORTION ',  !C03
;    0317  1                                                  'OF CRD !AC!/'),                                       !C03
;    0318  1        T_Contraction_Error                   = $ASCIC ('!/ERROR WHILE ATTEMPTING TO EXIT CRD !AC!/'),
;    0319  1        T_Incompatibility_Error               = $ASCIC ('!/INCORRECT VERSION OF CRD !AC!/'),             !C03
;    0320  1        T_Corrupted_Error                     = $ASCIC ('!/THE LOADABLE PORTION OF CRD !AC IS CORRUPTED!/'),     !C03
;  P 0321  1        T_MM_On                               = $ASCIC ('!/CANNOT CONTINUE CRD - MEMORY MANAGEMENT MUST BE OFF',
;  P 0322  1                                                  '!/TO TURN OFF MEMORY MANAGEMENT, TYPE "SET MM OFF"',
;    0323  1                                                  '!/RESTART CRD BY TYPING "CRD"!/'),
;    0324  1
;    0325  1        T_AUTO_Abort_CRD                      = $ASCIC ('!/!/AUTO TEST ABORTED - CALL FIELD SERVICE'),
;    0326  1        T_AUTO_End_Message                    = $ASCIC ('!/**** END OF VAX-11/730,725 AUTO TEST ****'),
;    0327  1        T_AUTO_Exit_To_Console                = $ASCIC ('!/!/RETURNING TO VAX-11/730,725 CONSOLE, WAIT FOR ">>>" PROMPT!/'
;    0328  1        T_No_DSA_Bit_Set                      = $ASCIC ('*** No DSA bit is set - DSA$GL_Flags = !XL(X)!/'),
;    0329  1
;    0330  1        T_Wrong_CPU                           = $ASCIC ('*** CRD cannot be run on this CPU - this is !AC CPU!/'),
;    0331  1        T_CPU_780                             = $ASCIC ('a VAX-11/780'),
;    0332  1        T_CPU_750                             = $ASCIC ('a VAX-11/750'),
;    0333  1        T_CPU_Unknown                         = $ASCIC ('an unknown'),
;    0334  1
;  P 0335  1        T_Determine_Size                      = $ASCIC ('*** Error while determining size of the ENSAB file ',
;    0336  1                                                  '- Status = !XL(X)!/'),
;    0337  1        T_Expand_Region                       = $ASCIC ('*** Error while expanding the region - Status = !XL(X)!/'),
;  P 0338  1        T_Load_File                           = $ASCIC ('*** Error while loading the ENSAB file into memory ',
;    0339  1                                                  '- Status = !XL(X)!/'),
;    0340  1
;    0341  1        T_CRD_Internal                        = $ASCIC ('*** A CRD Internal Error occurred!/'),
;    0342  1        T_CRD_Internal_Header                 = $ASCIC ('*** The following are the CRD debug vectors:!/'),
```

```
;   0343  1                 T_CRD_Internal_Debug            = $ASCIC ('*** Debug vector [!UL]: !XL(X)   !-!SL(D)!/'),
;   0344  1
;   0345  1                 T_Not_Contracted                = $ASCIC ('*** The region was not contracted - Status = !XL(X)!/'),
; P 0346  1                 T_Corrupted                     = $ASCIC ('*** Either the Null byte is not null, ',
;   0347  1                                                           'or the version numbers are wrong!/'),
;   0348  1                 T_Incompatible                  = $ASCIC ('*** These versions of ENSAB and ENSAA are incompatible!/'),
;   0349  1
;   0350  1
; P 0351  1                 T_Trace_Header                  = $ASCIC ('!/*** Beginning the CRD Tracing Facility',
; P 0352  1                                                           '!/*** All statements that begin with "***" ',
;   0353  1                                                           'are CRD trace statements!/!/'),
;   0354  1
;   0355  1
;   0356  1                 T_Trace_1                       = $ASCIC ('*** DSR$Load_CRD: Starting location of CRD: !XL(X)!/'),
;   0357  1                 T_Trace_2                       = $ASCIC ('*** DSR$Load_CRD: Size of the CRD image:    !XL(X)!/!/'),
;   0358  1
;   0359  1                 T_Trace_4                       = $ASCIC ('*** DSR$Load_CRD: Vector count:           DS = !4SB  CRD = !4SB!
;   0360  1                 T_Trace_5                       = $ASCIC ('*** DSR$Load_CRD: Positive version number: DS = !4SB  CRD = !4SB!
;   0361  1                 T_Trace_6                       = $ASCIC ('*** DSR$Load_CRD: Negative version number: DS = !4SB  CRD = !4SB!
;   0362  1                 T_Trace_7                       = $ASCIC ('*** DSR$Load_CRD: Null byte:              DS = !4SB  CRD = !4SB!
;   0363  1     %SBTTL  'Own Storage'
;   0364  1
;   0365  1     OWN
;   0366  1                 !+
;   0367  1                 ! An array containing the original values in the DSVECS dispatch vectors. Anytime the vectors starting at
;   0368  1                 ! DS$AL_DS_Dispatch_Vectors (DSVECS module) do not contain what they originally had, this array does contain
;   0369  1                 ! their original values. Otherwise, this array contains %X'FFFFFFFF' so as to provide a debugging check.
;   0370  1                 !-
;   0371  1
;   0372  1                 DS$A_CRD_Vectors                : VECTOR [CRD$K_Total_Numb_Vectors, LONG]                                !C07
;   0373  1                                                   INITIAL (REP CRD$K_Total_Numb_Vectors OF (%X'FFFFFFFF')),              !C07
;   0374  1
;   0375  1                 !+
;   0376  1                 ! A descriptor pointing to the DS$A_CRD_Vectors array. This descriptor is used to access the array.
;   0377  1                 !-
;   0378  1
;   0379  1                 DS$Q_CRD_Vector_Desc            : BLOCK [8, BYTE]
;   0380  1                                                   PRESET (
;   0381  1                                                         [DSC$W_Length]  = 0,
;   0382  1                                                         [DSC$A_Pointer] = 0,
;   0383  1                                                         [DSC$B_DType]   = 0,
;   0384  1                                                         [DSC$B_Class]   = 0
;   0385  1                                                   ),
;   0386  1
;   0387  1                 !+
;   0388  1                 ! A 'descriptor' for the CRD image itself.
;   0389  1                 !-
;   0390  1
;   0391  1                 DS$Q_CRD_Image_Desc             : VECTOR [2, LONG]
;   0392  1                                                   PRESET (
;   0393  1                                                         [K_Image_Size] = 0,
;   0394  1                                                         [K_Image_Ptr]  = 0
;   0395  1                                                   ),
;   0396  1
;   0397  1                 !+
;   0398  1                 ! Save the value of the DS$GA_LastAdr before it is changed in the Load_CRD routine.
;   0399  1                 !-
```

```
                DS$A_Saved_LastAdr              : INITIAL (0),

                !+
                ! Save the contents of the DSA$GL_Flags longword before starting CRD - for restoring it to it's
                ! original value. Do this for the DS flags also.
                !-

                DS$L_Saved_DSA_Flags            : INITIAL (0),
                DS$L_Saved_DS_Flags             : INITIAL (0),

                !+
                ! Points to either T_Menu, T_Auto, or T_Null.
                !-

                DS$A_Auto_Or_Menu;

        MACRO                                                                                          !EO7
                Module_Debug (Debug_String) =                                                          ! V
                        %IF %VARIANT
                        %THEN
                                !+
                                ! If this module has been compiled with the /VARIANT:n qualifier where n <> 0, then the foll
                                ! debugging statements will be compiled also. This can be invoked with the following:
                                !       Module_Debug ('A debug string');
                                !-
                                BEGIN
                                MAP
                                        DS$A_CRD_Vectors      : VECTOR [, LONG],
                                        DS$Q_CRD_Image_Desc   : VECTOR [, LONG],
                                        DS$Q_CRD_Vector_Desc  : VECTOR [, LONG];

                                BIND
                                        Debug_Out_1 = $ASCIC ('**** ', Debug_String, '!/'),
                                        Debug_Out_2 = $ASCIC ('****     DSA Flags        = !XL, DS Flags          = !XL!/'),
                                        Debug_Out_3 = $ASCIC ('****     Image_Desc  [0] = !XL, Image_Desc  [1] = !XL!/'),
                                        Debug_Out_4 = $ASCIC ('****     Vector_Desc [0] = !XL, Vector_Desc [1] = !XL!/'),
                                        Debug_Out_5 = $ASCIC ('****     Saved_Vector[0] = !XL, Saved_Vector[1] = !XL!/'),
                                        Debug_Out_6 = $ASCIC ('****     Saved_LastAdr   = !XL, DS$GA__LastAdr  = !XL!/');

                                IF (.DSA$V_CRD_MenuTest_Off OR
                                    .DSA$V_CRD_AutoTest_Off OR
                                    .DSA$V_CRD_MenuTest_On) THEN
                                                !                               [16]
                                        BEGIN
                                        $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_1);
                                        $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_2,
                                                .DSA$GL_Flags, .DS$GL_Flags);
                                        $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_3,
                                                .DS$Q_CRD_Image_Desc [0], .DS$Q_CRD_Image_Desc [1]);
                                        $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_4,
                                                .DS$Q_CRD_Vector_Desc [0], .DS$Q_CRD_Vector_Desc [1]);
                                        $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_5,
                                                .DS$A_CRD_Vectors [0], .DS$A_CRD_Vectors [1]);
                                        $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_6,
                                                .DS$A_Saved_LastAdr, .DS$GA_LastAdr);
                                        END
```

```
;  M 0457  1                            ELSE
;  M 0458  1                                    BEGIN
;  M 0459  1                                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_1);
;  M 0460  1                                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_2,
;  M 0461  1                                      .DSA$GL_Flags, .DS$GL_Flags);
;  M 0462  1                                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_3,
;  M 0463  1                                      .DS$Q_CRD_Image_Desc [0], .DS$Q_CRD_Image_Desc [1]);
;  M 0464  1                                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_4,
;  M 0465  1                                      .DS$Q_CRD_Vector_Desc [0], .DS$Q_CRD_Vector_Desc [1]);
;  M 0466  1                                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_5,
;  M 0467  1                                      .DS$A_CRD_Vectors [0], .DS$A_CRD_Vectors [1]);
;  M 0468  1                                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_6,
;  M 0469  1                                      .DS$A_Saved_LastAdr, .DS$GA_LastAdr);
;  M 0470  1                                    END
;  M 0471  1                            END
;  M 0472  1                    %FI                                                                  ! ^
;    0473  1                %;                                                                       !C07
;    0474  1        %SBTTL 'DSR$Load_CRD Routine'
;    0475  1
;    0476  1        GLOBAL ROUTINE DSR$Load_CRD : NOVALUE =
;    0477  1
;    0478  1        !++
;    0479  1        ! Functional Description:
;    0480  1        !
;    0481  1        !       Load the CRD Image file into memory starting at the last address in the Supervisor. This address
;    0482  1        !       is contained in the variable DS$GA_LastAdr. Reset this variable after CRD has been loaded in to point
;    0483  1        !       to the last address used by CRD. Expand the region occupied by the Supervisor.
;    0484  1        !
;    0485  1        ! Formal Parameters:
;    0486  1        !
;    0487  1        !       None
;    0488  1        !
;    0489  1        ! Side Effects:
;    0490  1        !
;    0491  1        !       Changes the address contained in DS$GA_LastAdr.
;    0492  1        !       Loads the CRD image file into memory starting at the last address used by the Resident-DS.
;    0493  1        !       Exchange dispatch vectors with CRD and DS for the CRD routines.
;    0494  1        !
;    0495  1        ! Completion Codes:
;    0496  1        !
;    0497  1        !       None
;    0498  1        !--
;    0499  2        BEGIN                                           ! Routine DSR$Load_CRD
;    0500  2                MAP
;    0501  2                        DSA$GL_SID : VECTOR [, BYTE];
;    0502  2
;    0503  2                REGISTER
;    0504  2                        Page_Count,                     ! The number of pages of memory required for CRD image
;    0505  2                        Status;                         ! A status variable
;    0506  2
;    0507  2                LOCAL
;    0508  2                        CRD_Actual_Length,              ! The actual length of the file in bytes
;    0509  2                        CRD_File_Attributes,            ! The attributes of the CRD image file
;    0510  2
;    0511  2                        CRD_Image_Location              : VECTOR [2, LONG];
;    0512  2                                                        ! The starting and ending addresses of the expanded region
;    0513  2
```

```
;    0514  2              Module_Debug ('DSR$Load_CRD - Start:');                                                    !C07
;    0515  2
;    0516  2              IF (NOT $DS_MMOff) THEN
;    0517  2                      CRD_Error (K_MM_Is_On, 0, K_Region_Not_Expanded);
;    0518  2
;    0519  2              DS$L_Saved_DSA_Flags = .DSA$GL_Flags;
;    0520  2              DS$L_Saved_DS_Flags  = .DS$GL_Flags;
;    0521  2
;    0522  2              !+
;    0523  2              ! First, make sure that the Binary flag is set so that any subsequent $Print's with the
;    0524  2              ! CRD_AutoTest typecode won't get the first byte of the output string cut off (with Binary
;    0525  2              ! set, the TypeCode is stuck onto the front of the output string, and then if the TypeCode
;    0526  2              ! is CRD_AutoTest, then this first byte gets chopped off - by DSX$TypeOut in the PRINT module).
;    0527  2              !-
;    0528  2
;    0529  2              DSA$V_Binary = On;
;    0530  2
;    0531  2              IF (.DSA$V_CRD_Trace) THEN
;    0532  2                      $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_Header);
;    0533  2
;    0534  2              !+
;    0535  2              ! If the Debug longword is set (via the SET CRD/DEBUG/TRACE command), set the Debug flag bit.   !C07]
;    0536  2              ! This has to be done because the CRD_Error routine has to check a debug flag to see if the    !C07]
;    0537  2              ! Internal_Error routine should be called. However, at the time this check is done, the        !C07]
;    0538  2              ! CRD_Debug longword has the address of the start of the CRD image (due to the transfer of     !C07]
;    0539  2              ! vectors). So, we use another flag to say that debug is on for this one case only.           !C07]
;    0540  2              !-
;    0541  2
;    0542  2              IF (.DS$GL_CRD_Debug) THEN                                                                   !C07]
;    0543  2                      CRD$V_CRD_Debug = True;                                                              !C07]
;    0544  2
;    0545  2              IF (.DSA$V_CRD_AutoTest_Off) THEN
;    0546  2                      DS$A_Auto_Or_Menu = T_Auto
;    0547  2              ELSE IF (.DSA$V_CRD_MenuTest_Off OR .DSA$V_CRD_MenuTest_On) THEN
;    0548  2                                      !                                                    C10]
;    0549  3                      BEGIN                                                                                !C06]
;    0550  3                      DS$A_Auto_Or_Menu = T_Menu;                                                          !C06]
;    0551  3                      CRD$V_Ctrl_C_No_Echo = True;                                                         !C07]
;    0552  3                      CRD$V_Flush_Ctrl_C = False;                                                          !C07]
;    0553  3                      END                                                                                  !C06]
;    0554  2              ELSE
;    0555  3                      BEGIN
;    0556  3                      DS$A_Auto_Or_Menu = T_Null;
;    0557  3                      CRD_Error (K_No_DSA_Bit_Set, .DSA$GL_Flags, K_Region_Not_Expanded);
;    0558  2                      END;
;    0559  2
;    0560  2              !+
;    0561  2              ! Check to see if this is a correct CPU for running CRD.
;    0562  2              !-
;    0563  2
;    0564  2              IF (.DSA$GL_SID [3] NEQ SID_730) THEN
;    0565  2                      CRD_Error (K_Wrong_CPU, .DSA$GL_SID [3], K_Region_Not_Expanded);
;    0566  2
;    0567  2              IF (.DS$Q_CRD_Vector_Desc [DSC$W_Length] EQL 0) THEN
;    0568  3                      BEGIN
;    0569  3                      MAP
;    0570  3                              DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE, SIGNED];
```

```
;    0571  3
;    0572  3                    !+
;    0573  3                    ! These stored values will later be put back into the dispatch area when CRD exits. Store all
;    0574  3                    ! the vectors plus the 4 bytes at the beginning of the dispatch vectors. Store a pointer to the
;    0575  3                    ! block, and the ACTUAL size of the block in a descriptor.
;    0576  3                    !-
;    0577  3
;    0578  3                    DS$Q_CRD_Vector_Desc [DSC$W_Length] = CRD$K_Total_Numb_Vectors * 4;                          !E07
;    0579  3                    DS$Q_CRD_Vector_Desc [DSC$A_Pointer] = DS$A_CRD_Vectors;
;    0580  3
;    0581  3                    !+
;    0582  3                    ! Now transfer the four bytes at the beginning (version numbers, etc.) and the actual
;    0583  3                    ! dispatch vectors to the temporary storage.
;    0584  3                    !-
;    0585  3
;    0586  3                    INCR Vector_Index FROM 0 TO (.DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors] - 1) DO        !E07
;    0587  4                            BEGIN
;    0588  4                            MAP
;    0589  4                                    DS$AL_DS_Dispatch_Vectors : VECTOR [, LONG];
;    0590  4
;    0591  4                            BIND
;    0592  4                                    Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, LONG];
;    0593  4
;    0594  4                            Temp_Vectors [.Vector_Index] = .DS$AL_DS_Dispatch_Vectors [.Vector_Index];
;    0595  4                            END
;    0596  3                    END
;    0597  2            ELSE
;    0598  2                    Module_Debug ('DSR$Load_CRD: ***** ERROR *****');
;    0599  2
;    0600  2            !+
;    0601  2            ! Now do a 'fake' Load of the CRD image. This will return the size of the image, in
;    0602  2            ! preparation for allocating the necessary memory and loading the image into memory.
;    0603  2            !-
;    0604  2
;  P 0605  4            IF (NOT (Status = $DS_LOAD (
;  P 0606  4                              File    = T_CRD_Image_Name,
;  P 0607  4                              Default = T_CRD_Image_Default,
;  P 0608  4                              Length  = 0,
;  P 0609  4                              Address = 0,
;  P 0610  4                              RetLen  = CRD_Actual_Length,
;  P 0611  4                              RetRec  = CRD_File_Attributes,
;    0612  3                              VBN     = 1)))
;    0613  2            THEN
;    0614  2                    CRD_Error (K_Determine_Size, .Status, K_Region_Not_Expanded);
;    0615  2
;    0616  2            DS$A_Saved_LastAdr = .DS$GA_LastAdr;
;    0617  2                                            ! Save the old value.
;    0618  2
;    0619  2            Page_Count = (.CRD_Actual_Length + 511) / 512;
;    0620  2                                            ! Number of pages required to load CRD
;    0621  2
;    0622  2            IF (.DSA$V_User) THEN                                                                               !E04
;    0623  3                    BEGIN                                                                                       !E04
;    0624  3                    !+
;    0625  3                    ! Use this one for running on-line.
;    0626  3                    !-
;  P 0627  3                    Status = $ExpReg
```

```
;  P 0628  3                                (
;  P 0629  3                                PagCnt = .Page_Count,
;  P 0630  3                                RetAdr = CRD_Image_Location,
;  P 0631  3                                Region = 0
;    0632  4                                )       ! Expand the Supervisor region by the number of pages
;    0633  4                                        ! required to hold the CRD image, set the access mode to
;    0634  4                                        ! User-Write, expand P0 space.
;    0635  3                        END                                                                          !E04
;    0636  2                ELSE                                                                                 !E04
;    0637  3                        BEGIN                                                                        !E04
;    0638  3                        !+
;    0639  3                        ! Use this one for running stand-alone.
;    0640  3                        !-
;    0641  3                        Status = CRD$ExpReg
;    0642  3                                (
;    0643  3                                .Page_Count,
;    0644  3                                CRD_Image_Location,
;    0645  3                                0,
;    0646  3                                3
;    0647  3                                );      ! Expand the Supervisor region by the number of pages
;    0648  3                                        ! required to hold the CRD image, use no access mode,
;    0649  3                                        ! expand DS space (3).
;    0650  3                        END;                                                                         !E04
;    0651  2
;    0652  2                DS$Q_CRD_Image_Desc [K_Image_Size] =
;    0653  2                        .CRD_Image_Location [K_Ending_Adr] - .CRD_Image_Location [K_Starting_Adr] + 1;
;    0654  2
;    0655  2                DS$Q_CRD_Image_Desc [K_Image_Ptr] = .CRD_Image_Location [K_Starting_Adr];
;    0656  2
;    0657  2                IF (NOT .Status) THEN
;    0658  2                        !+
;    0659  2                        ! If an error occurred, check the return address array. If both addresses are -1, then
;    0660  2                        ! no memory was expanded. If they are not both zero, some was expanded.
;    0661  2                        !-
;    0662  3                        IF ((.CRD_Image_Location [K_Starting_Adr] EQL -1) AND
;    0663  3                            (.CRD_Image_Location [K_Ending_Adr]   EQL -1))
;    0664  2                        THEN
;    0665  3                                BEGIN
;    0666  3                                DS$Q_CRD_Image_Desc [K_Image_Size] = 0;
;    0667  3                                DS$Q_CRD_Image_Desc [K_Image_Ptr]  = 0;
;    0668  3
;    0669  3                                CRD_Error (K_Expand_Region, .Status, K_Region_Not_Expanded);
;    0670  3                                END
;    0671  2                        ELSE
;    0672  2                                CRD_Error (K_Expand_Region, .Status, K_Region_Expanded);
;    0673  2
;  P 0674  2                Status = $DS_LOAD (
;  P 0675  2                                   File    = T_CRD_Image_Name,                                       !E03
;  P 0676  2                                   Default = T_CRD_Image_Default,                                     !E03
;  P 0677  2                                   Length  = .CRD_Actual_Length,
;  P 0678  2                                   Address = .CRD_Image_Location [K_Starting_Adr],
;  P 0679  2                                   RetLen  = CRD_Actual_Length,
;  P 0680  2                                   RetRec  = CRD_File_Attributes,
;  P 0681  2                                   VBN     = 1
;    0682  2                                  );          ! Actually load the file into the expanded region
;    0683  2
;    0684  2                IF (NOT .Status) THEN
```

```
;    0685  2                    CRD_Error (K_Load_File, .Status, K_Region_Expanded);
;    0686  2
;    0687  2            DS$GA_LastAdr = .DS$GA_LastAdr + .CRD_Actual_Length;
;    0688  2                                    ! Reset the LastAdr pointer
;    0689  2
;    0690  2            IF (.DSA$V_CRD_Trace) THEN
;    0691  3                BEGIN
;    0692  3                MAP
;    0693  3                    DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE, SIGNED];
;    0694  3
;    0695  3                BIND
;    0696  3                    CRD_Bytes = (.CRD_Image_Location [K_Starting_Adr]) : VECTOR [, BYTE, SIGNED];
;    0697  3
;    0698  3                $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_1, .DS$Q_CRD_Image_Desc [K_Image_Ptr]);
;    0699  3                $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_2, .DS$Q_CRD_Image_Desc [K_Image_Size]);
;    0700  3
;  P 0701  3                $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_4,
;  P 0702  3                    .DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors],
;    0703  3                    .CRD_Bytes [K_Number_Of_Vectors]);
;    0704  3
;  P 0705  3                $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_5,
;  P 0706  3                    .DS$AL_DS_Dispatch_Vectors [K_Positive_Version],
;    0707  3                    .CRD_Bytes [K_Positive_Version]);
;    0708  3
;  P 0709  3                $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_6,
;  P 0710  3                    .DS$AL_DS_Dispatch_Vectors [K_Negative_Version],
;    0711  3                    .CRD_Bytes [K_Negative_Version]);
;    0712  3
;  P 0713  3                $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_7,
;  P 0714  3                    .DS$AL_DS_Dispatch_Vectors [K_Null_Byte],
;    0715  3                    .CRD_Bytes [K_Null_Byte]);
;    0716  2                END;
;    0717  2
;    0718  2            Exchange_Dispatch_Vectors (.CRD_Image_Location [K_Starting_Adr]);                    !C03
;    0719  2                                    ! Exchange all the dispatch vectors for CRD and DS
;    0720  2
;    0721  2            Module_Debug ('DSR$Load_CRD - End:');                                               !C07
;    0722  1    END;                            ! Routine DSR$Load_CRD


                                                .TITLE  CRDIMAGE *** Loading and Unloading the CRD Imag
                                                                 e
;                                               .IDENT  \02-11\

                                                .PSECT  WORK,NOEXE,2

                                FFFFFFFF# 00000 DS$A_CRD_VECTORS:
                                                .LONG   -1[25]                                          ;
                                    0000  00064 DS$Q_CRD_VECTOR_DESC:
                                                .WORD   0                                               ;
                                  00  00  00066         .BYTE   0, 0                                    ;
                                00000000  00068         .LONG   0                                       ;
                            00000000  00000000  0006C DS$Q_CRD_IMAGE_DESC:
                                                .LONG   0, 0                                            ;
                                00000000  00074 DS$A_SAVED_LASTADR:
                                                .LONG   0                                               ;
                                00000000  00078 DS$L_SAVED_DSA_FLAGS:
```

```
                                                              .LONG    0                                               ;
                                    00000000  0007C DS$L_SAVED_DS_FLAGS:
                                                              .LONG    0                                               ;
                                              00080 DS$A_AUTO_OR_MENU:
                                                              .BLKB    4

                                                              .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                  45 47 41 4D 49 44 52 43 08  00000 P.AAA:    .ASCII  <8>\CRDIMAGE\                                   ;
            30 3B 45 58 45 2E 42 41 53 4E 45  00009 P.AAC:    .ASCII  \ENSAB.EXE;0\                                   ;
                                   0000000B   00014 P.AAB:    .LONG   11                                              ;
                                   00000000'  00018           .ADDRESS P.AAC                                          ;
                                              0001C P.AAE:    .BLKB   0                                               ;
                                   00000000   0001C P.AAD:    .LONG   0                                               ;
                                   00000000'  00020           .ADDRESS P.AAE                                          ;
                                         00   00024 P.AAF:    .ASCII  <0>                                             ;
               54 53 45 54 20 55 4E 45 4D 09  00025 P.AAG:    .ASCII  <9>\MENU TEST\                                  ;
               54 53 45 54 20 4F 54 55 41 09  0002F P.AAH:    .ASCII  <9>\AUTO TEST\                                  ;
59 4C 4E 4F 20 43 41 21 20 44 52 43 2F 21 2E  00039 P.AAI:    .ASCII  \.!/CRD !AC ONLY SUPPORTED ON A VAX-11/73\      ;
41 20 4E 4F 20 44 45 54 52 4F 50 50 55 53 20  00048                                                                   ;
               33 37 2F 31 31 2D 58 41 56 20  00057                                                                   ;
                        2F 21 35 32 37 2C 30  00061           .ASCII  \0,725!/\                                       ;
49 46 20 4F 54 20 45 4C 42 41 4E 55 2F 21 32  00068 P.AAJ:    .ASCII  \2!/UNABLE TO FIND THE LOADABLE PORTION O\      ;
45 4C 42 41 44 41 4F 4C 20 45 48 54 20 44 4E  00077                                                                   ;
               4F 20 4E 4F 49 54 52 4F 50 20  00086                                                                   ;
            2F 21 43 41 21 20 44 52 43 20 46  00090           .ASCII  \F CRD !AC!/\                                   ;
20 45 4C 49 48 57 20 52 4F 52 52 45 2F 21 42  0009B P.AAK:    .ASCII  \B!/ERROR WHILE ATTEMPTING TO LOAD THE LO\      ;
4C 20 4F 54 20 47 4E 49 54 50 4D 45 54 54 41  000AA                                                                   ;
               4F 4C 20 45 48 54 20 44 41 4F  000B9                                                                   ;
20 4E 4F 49 54 52 4F 50 20 45 4C 42 41 44 41  000C3           .ASCII  \ADABLE PORTION OF CRD !AC!/\                   ;
         2F 21 43 41 21 20 44 52 43 20 46 4F  000D2                                                                   ;
20 45 4C 49 48 57 20 52 4F 52 52 45 2F 21 2A  000DE P.AAL:    .ASCII  \*!/ERROR WHILE ATTEMPTING TO EXIT CRD !A\      ;
45 20 4F 54 20 47 4E 49 54 50 4D 45 54 54 41  000ED                                                                   ;
               41 21 20 44 52 43 20 54 49 58  000FC                                                                   ;
                                    2F 21 43  00106           .ASCII  \C!/\                                           ;
45 56 20 54 43 45 52 52 4F 43 4E 49 2F 21 20  00109 P.AAM:    .ASCII  \ !/INCORRECT VERSION OF CRD !AC!/\             ;
41 21 20 44 52 43 20 46 4F 20 4E 4F 49 53 52  00118                                                                   ;
                                    2F 21 43  00127                                                                   ;
45 4C 42 41 44 41 4F 4C 20 45 48 54 2F 21 30  0012A P.AAN:    .ASCII  \0!/THE LOADABLE PORTION OF CRD !AC IS CO\      ;
44 52 43 20 46 4F 20 4E 4F 49 54 52 4F 50 20  00139                                                                   ;
               4F 43 20 53 49 20 43 41 21 20  00148                                                                   ;
                  2F 21 44 45 54 50 55 52 52  00152           .ASCII  \RRUPTED!/\                                     ;
49 54 4E 4F 43 20 54 4F 4E 4E 41 43 2F 21 86  0015B P.AAO:    .ASCII  <134>\!/CANNOT CONTINUE CRD - MEMORY MAN\       ;
52 4F 4D 45 4D 20 2D 20 44 52 43 20 45 55 4E  0016A                                                                   ;
                                 4E 41 4D 20 59  00179                                                                ;
45 42 20 54 53 55 4D 20 54 4E 45 4D 45 47 41  0017E           .ASCII  \AGEMENT MUST BE OFF!/TO TURN OFF MEMORY \      ;
4F 20 4E 52 55 54 20 4F 54 2F 21 46 46 4F 20  0018D                                                                   ;
               20 59 52 4F 4D 45 4D 20 46 46  0019C                                                                   ;
50 59 54 20 2C 54 4E 45 4D 45 47 41 4E 41 4D  001A6           .ASCII  \MANAGEMENT, TYPE "SET MM OFF"!/RESTART C\      ;
21 22 46 46 4F 20 4D 4D 20 54 45 53 22 20 45  001B5                                                                   ;
               43 20 54 52 41 54 53 45 52 2F  001C4                                                                   ;
43 22 20 47 4E 49 50 59 54 20 59 42 20 44 52  001CE           .ASCII  \RD BY TYPING "CRD"!/\                          ;
                           2F 21 22 44 52     001DD                                                                   ;
20 54 53 45 54 20 4F 54 55 41 2F 21 2F 21 2A  001E2 P.AAP:    .ASCII  \*!/!/AUTO TEST ABORTED - CALL FIELD SERV\      ;
20 4C 4C 41 43 20 2D 20 44 45 54 52 4F 42 41  001F1                                                                   ;
               56 52 45 53 20 44 4C 45 49 46  00200                                                                   ;
                                    45 43 49  0020A           .ASCII  \ICE\                                           ;
```

```
20 46 4F 20 44 4E 45 20 2A 2A 2A 2A 2F 21 2B 0020D P.AAQ:  .ASCII  \+!/**** END OF VAX-11/730,725 AUTO TEST \
20 35 32 37 2C 30 33 37 2F 31 31 2D 58 41 56 0021C
                20 54 53 45 54 20 4F 54 55 41 0022B
                                  2A 2A 2A 2A 00235          .ASCII  \****\
20 47 4E 49 4E 52 55 54 45 52 2F 21 2F 21 40 00239 P.AAR:  .ASCII  \@!/!/RETURNING TO VAX-11/730,725 CONSOLE\
37 2C 30 33 37 2F 31 31 2D 58 41 56 20 4F 54 00248
                45 4C 4F 53 4E 4F 43 20 35 32 00257
3E 3E 3E 22 20 52 4F 46 20 54 49 41 57 20 2C 00261          .ASCII  \, WAIT FOR ">>>" PROMPT!/\
                2F 21 54 50 4D 4F 52 50 20 22 00270
74 69 62 20 41 53 44 20 6F 4E 20 2A 2A 2A 2F 0027A P.AAS:  .ASCII  \/*** No DSA bit is set - DSA$GL_Flags = \
47 24 41 53 44 20 2D 20 74 65 73 20 73 69 20 00289
                20 3D 20 73 67 61 6C 46 5F 4C 00298
                   2F 21 29 58 28 4C 58 21 002A2          .ASCII  \!XL(X)!/\
74 6F 6E 6E 61 63 20 44 52 43 20 2A 2A 2A 35 002AA P.AAT:  .ASCII  \5*** CRD cannot be run on this CPU - thi\
73 69 68 74 20 6E 6F 20 6E 75 72 20 65 62 20 002B9
                69 68 74 20 2D 20 55 50 43 20 002C8
      2F 21 55 50 43 20 43 41 21 20 73 69 20 73 002D2          .ASCII  \s is !AC CPU!/\
         30 38 37 2F 31 31 2D 58 41 56 20 61 0C 002E0 P.AAU:  .ASCII  <12>\a VAX-11/780\
         30 35 37 2F 31 31 2D 58 41 56 20 61 0C 002ED P.AAV:  .ASCII  <12>\a VAX-11/750\
                6E 77 6F 6E 6B 6E 75 20 6E 61 0A 002FA P.AAW:  .ASCII  <10>\an unknown\
6C 69 68 77 20 72 6F 72 72 45 20 2A 2A 2A 46 00305 P.AAX:  .ASCII  \F*** Error while determining size of the\
73 20 67 6E 69 6E 69 6D 72 65 74 65 64 20 65 00314
                65 68 74 20 66 6F 20 65 7A 69 00323
53 20 2D 20 65 6C 69 66 20 42 41 53 4E 45 20 0032D          .ASCII  \ ENSAB file - Status = !XL(X)!/\
21 29 58 28 4C 58 21 20 3D 20 73 75 74 61 74 0033C
                                           2F 0034B
6C 69 68 77 20 72 6F 72 72 45 20 2A 2A 2A 38 0034C P.AAY:  .ASCII  \8*** Error while expanding the region - \
65 68 74 20 67 6E 69 64 6E 61 70 78 65 20 65 0035B
                20 2D 20 6E 6F 69 67 65 72 20 0036A
29 58 28 4C 58 21 20 3D 20 73 75 74 61 74 53 00374          .ASCII  \Status = !XL(X)!/\
                                        2F 21 00383
6C 69 68 77 20 72 6F 72 72 45 20 2A 2A 2A 46 00385 P.AAZ:  .ASCII  \F*** Error while loading the ENSAB file \
45 20 65 68 74 20 67 6E 69 64 61 6F 6C 20 65 00394
                20 65 6C 69 66 20 42 41 53 4E 003A3
53 20 2D 20 79 72 6F 6D 65 6D 20 6F 74 6E 69 003AD          .ASCII  \into memory - Status = !XL(X)!/\
21 29 58 28 4C 58 21 20 3D 20 73 75 74 61 74 003BC
                                           2F 003CB
65 74 6E 49 20 44 52 43 20 41 20 2A 2A 2A 23 003CC P.ABA:  .ASCII  \#*** A CRD Internal Error occurred!/\
75 63 63 6F 20 72 6F 72 72 45 20 6C 61 6E 72 003DB
                   2F 21 64 65 72 72 003EA
77 6F 6C 6C 6F 66 20 65 68 54 20 2A 2A 2A 2E 003F0 P.ABB:  .ASCII  \.*** The following are the CRD debug vec\
44 52 43 20 65 68 74 20 65 72 61 20 67 6E 69 003FF
                63 65 76 20 67 75 62 65 64 20 0040E
                   2F 21 3A 73 72 6F 74 00418          .ASCII  \tors:!/\
74 63 65 76 20 67 75 62 65 44 20 2A 2A 2A 2B 0041F P.ABC:  .ASCII  \+*** Debug vector [!UL]: !XL(X)   !~!SL(\
58 28 4C 58 21 20 3A 5D 4C 55 21 5B 20 72 6F 0042E
                28 4C 53 21 2D 21 20 20 20 29 0043D
                                  2F 21 29 44 00447          .ASCII  \D)!/\
6E 6F 69 67 65 72 20 65 68 54 20 2A 2A 2A 35 0044B P.ABD:  .ASCII  \5*** The region was not contracted - Sta\
61 72 74 6E 6F 63 20 74 6F 6E 20 73 61 77 20 0045A
                61 74 53 20 2D 20 64 65 74 63 00469
      2F 21 29 58 28 4C 58 21 20 3D 20 73 75 74 00473          .ASCII  \tus = !XL(X)!/\
65 68 74 20 72 65 68 74 69 45 20 2A 2A 2A 48 00481 P.ABE:  .ASCII  \H*** Either the Null byte is not null, o\
6E 20 73 69 20 65 74 79 62 20 6C 6C 75 4E 20 00490
                6F 20 2C 6C 6C 75 6E 20 74 6F 0049F
6E 20 6C 6F 69 73 72 65 76 20 65 68 74 20 72 004A9          .ASCII  \r the version numbers are wrong!/\
6E 6F 72 77 20 65 72 61 20 73 72 65 62 6D 75 004B8
```

```
                                                                                                2F 21 67 004C7
73 72 65 76 20 65 73 65 68 54 20 2A 2A 2A 38 004CA P.ABF:  .ASCII  \8*** These versions of ENSAB and ENSAA a\
61 20 42 41 53 4E 45 20 66 6F 20 73 6E 6F 69 004D9
                  61 20 41 41 53 4E 45 20 64 6E 004E8
65 6C 62 69 74 61 70 6D 6F 63 6E 69 20 65 72 004F2          .ASCII  \re incompatible!/\
                                                                2F 21 00501
6E 69 6E 6E 69 67 65 42 20 2A 2A 2A 2F 21 6F 00503 P.ABG:  .ASCII  \o!/*** Beginning the CRD Tracing Facilit\
69 63 61 72 54 20 44 52 43 20 65 68 74 20 67 00512
                  74 69 6C 69 63 61 46 20 67 6E 00521
74 61 74 73 20 6C 6C 41 20 2A 2A 2A 2F 21 79 0052B          .ASCII  \y!/*** All statements that begin with "*\
67 65 62 20 74 61 68 74 20 73 74 6E 65 6D 65 0053A
                  2A 22 20 68 74 69 77 20 6E 69 00549
61 72 74 20 44 52 43 20 65 72 61 20 22 2A 2A 00553          .ASCII  \**" are CRD trace statements!/!/\
2F 21 73 74 6E 65 6D 65 74 61 74 73 20 65 63 00562
                                                                2F 21 00571
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 34 00573 P.ABH:  .ASCII  \4*** DSR$Load_CRD: Starting location of \
6F 6C 20 67 6E 69 74 72 61 74 53 20 3A 44 52 00582
                  20 66 6F 20 6E 6F 69 74 61 63 00591
         2F 21 29 58 28 4C 58 21 20 3A 44 52 43 0059B          .ASCII  \CRD: !XL(X)!/\
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 36 005A8 P.ABI:  .ASCII  \6*** DSR$Load_CRD: Size of the CRD image\
65 68 74 20 66 6F 20 65 7A 69 53 20 3A 44 52 005B7
                  65 67 61 6D 69 20 44 52 43 20 005C6
2F 21 2F 21 29 58 28 4C 58 21 20 20 20 20 3A 005D0          .ASCII  \:    !XL(X)!/!/\
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42 005DF P.ABJ:  .ASCII  \B*** DSR$Load_CRD: Vector count:        \
6E 75 6F 63 20 72 6F 74 63 65 56 20 3A 44 52 005EE
                  20 20 20 20 20 20 20 20 3A 74 005FD
20 20 42 53 34 21 20 3D 20 53 44 20 20 20 20 00607          .ASCII  \    DS = !4SB  CRD = !4SB!/\
         2F 21 42 53 34 21 20 3D 20 44 52 43 00616
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42 00622 P.ABK:  .ASCII  \B*** DSR$Load_CRD: Positive version numb\
65 76 20 65 76 69 74 69 73 6F 50 20 3A 44 52 00631
                  62 6D 75 6E 20 6E 6F 69 73 72 00640
20 20 42 53 34 21 20 3D 20 53 44 20 3A 72 65 0064A          .ASCII  \er: DS = !4SB  CRD = !4SB!/\
         2F 21 42 53 34 21 20 3D 20 44 52 43 00659
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42 00665 P.ABL:  .ASCII  \B*** DSR$Load_CRD: Negative version numb\
65 76 20 65 76 69 74 61 67 65 4E 20 3A 44 52 00674
                  62 6D 75 6E 20 6E 6F 69 73 72 00683
20 20 42 53 34 21 20 3D 20 53 44 20 3A 72 65 0068D          .ASCII  \er: DS = !4SB  CRD = !4SB!/\
         2F 21 42 53 34 21 20 3D 20 44 52 43 0069C
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42 006A8 P.ABM:  .ASCII  \B*** DSR$Load_CRD: Null byte:          \
20 3A 65 74 79 62 20 6C 6C 75 4E 20 3A 44 52 006B7
                  20 20 20 20 20 20 20 20 20 20 006C6
20 20 42 53 34 21 20 3D 20 53 44 20 20 20 20 006D0          .ASCII  \    DS = !4SB  CRD = !4SB!/\
         2F 21 42 53 34 21 20 3D 20 44 52 43 006DF
```

```
DSA$AL_APTMAIL=     65024
DSA$GL_FLAGS=       65024
DSA$GL_APTCOM=      65028
DSA$GL_PASSES=      65032
DSA$GL_UNITS=       65036
DSA$GL_SECTNO=      65040
DSA$GL_SID=         65044
DSA$GL_MSGTYP=      65088
DSA$GL_ERRNO=       65092
DSA$GL_EVENT=       65096
DSA$GL_SUBTNO=      65100
DSA$GL_TESTNO=      65104
DSA$GL_PASSNO=      65108
```

```
DSA$GL_DEVLEN=      65112
DSA$GL_DEVNAM=      65116
DSA$GL_MSGPTR=      65128
$MODULE=            P.AAA
T_CRD_IMAGE_NAME=   P.AAB
T_CRD_IMAGE_DEFAULT=P.AAD
T_NULL=             P.AAF
T_MENU=             P.AAG
T_AUTO=             P.AAH
T_INVALID_CPU=      P.AAI
T_FILE_NOT_FOUND=   P.AAJ
T_UNKNOWN_ERROR=    P.AAK
T_CONTRACTION_ERROR=P.AAL
T_INCOMPATIBILITY_ERROR=
                    P.AAM
T_CORRUPTED_ERROR=  P.AAN
T_MM_ON=            P.AAO
T_AUTO_ABORT_CRD=   P.AAP
T_AUTO_END_MESSAGE= P.AAQ
T_AUTO_EXIT_TO_CONSOLE=
                    P.AAR
T_NO_DSA_BIT_SET=   P.AAS
T_WRONG_CPU=        P.AAT
T_CPU_780=          P.AAU
T_CPU_750=          P.AAV
T_CPU_UNKNOWN=      P.AAW
T_DETERMINE_SIZE=   P.AAX
T_EXPAND_REGION=    P.AAY
T_LOAD_FILE=        P.AAZ
T_CRD_INTERNAL=     P.ABA
T_CRD_INTERNAL_HEADER=
                    P.ABB
T_CRD_INTERNAL_DEBUG=
                    P.ABC
T_NOT_CONTRACTED=   P.ABD
T_CORRUPTED=        P.ABE
T_INCOMPATIBLE=     P.ABF
T_TRACE_HEADER=     P.ABG
T_TRACE_1=          P.ABH
T_TRACE_2=          P.ABI
T_TRACE_4=          P.ABJ
T_TRACE_5=          P.ABK
T_TRACE_6=          P.ABL
T_TRACE_7=          P.ABM
        .EXTRN  DS_CLEANUP, MAPFREE
        .EXTRN  CRD$EXPREG, DSR$CHECK_MENUTEST_OFF_SET
        .EXTRN  DSR$CHECK_AUTOTEST_OFF_SET
        .EXTRN  EXE$ALONONPAGED
        .EXTRN  EXE$DEANONPAGED
        .EXTRN  DSV$EXIT, BEGIN_BLISS
        .EXTRN  DS$GA_DS_CTRL_C_FIRST
        .EXTRN  DS$GA_DS_CTRL_C_SECOND
        .EXTRN  DS$L_USERCNTRLC
        .EXTRN  DS$GL_FLAGS, DS$AL_DS_DISPATCH_VECTORS
        .EXTRN  DS$GL_CRD_FLAGS
        .EXTRN  DS$GA_LASTADR, DS$GL_CRD_DEBUG
        .EXTRN  DS$MMOFF, DSX$PRINT
```

```
                                                        .EXTRN  DS$LOAD, SYS$EXPREG

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                OFFC 00000      .ENTRY  DSR$LOAD_CRD, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0476
                                                                        R10,R11
                                        5B 00000000G    9F  9E 00002         MOVAB   @#DS$LOAD, R11                  ;
                                        5A 00000000G    EF  9E 00009         MOVAB   DS$GL_CRD_FLAGS, R10            ;
                                        59 00000000G    EF  9E 00010         MOVAB   DS$AL_DS_DISPATCH_VECTORS, R9   ;
                                        58 00000000V    EF  9E 00017         MOVAB   CRD_ERROR, R8                   ;
                                        57 00000000G    9F  9E 0001E         MOVAB   @#DSX$PRINT, R7                 ;
                                        56 00000000'    EF  9E 00025         MOVAB   T_CRD_IMAGE_DEFAULT, R6         ;
                                        55 00000000'    EF  9E 0002C         MOVAB   DS$A_AUTO_OR_MENU, R5           ;
                                                5E      10  C2 00033         SUBL2   #16, SP                         ;
                                00000000G       9F      00  FB 00036         CALLS   #0, @#DS$MMOFF                  ; 0516
                                                08      50  E8 0003D         BLBS    R0, 1$                          ;
                                                        01  DD 00040         PUSHL   #1                              ; 0517
                                                7E      09  7D 00042         MOVQ    #9, -(SP)                       ;
                                                68      03  FB 00045         CALLS   #3, CRD_ERROR                   ;
                                    F8  A5 0000FE00     9F  D0 00048 1$:     MOVL    @#^X0000FE00, DS$L_SAVED_DSA_FLAGS ; 0519
                                    FC  A5 00000000G    EF  D0 00050         MOVL    DS$GL_FLAGS, DS$L_SAVED_DS_FLAGS ; 0520
                                0000FE00    9F          02  88 00058         BISB2   #2, @#^X0000FE00                ; 0529
                    54 0000FE02     9F                  01  EF 0005F         EXTZV   #1, #1, @#^X0000FE02, R4        ; 0531
                                                08      54  E9 00068         BLBC    R4, 2$                          ;
                                                    04E7    C6  9F 0006B         PUSHAB  T_TRACE_HEADER                  ; 0532
                                                            01  DD 0006F         PUSHL   #T                          ;
                                                            1A  DD 00071         PUSHL   #26                         ;
                                                67          03  FB 00073         CALLS   #3, DSX$PRINT                   ;
                                                03 00000000G    EF  E9 00076 2$:     BLBC    DS$GL_CRD_DEBUG, 3$     ; 0542
                                                6A          01  88 0007D         BISB2   #1, DS$GL_CRD_FLAGS         ; 0543
                        06 0000FE01             9F          03  E1 00080 3$:     BBC     #3, @#^X0000FE01, 4$    ; 0545
                                                65      13      A6  9E 00088         MOVAB   T_AUTO, DS$A_AUTO_OR_MENU   ; 0546
                                                            2C  11 0008C         BRB     7$                          ;
                                                08 0000FE02     9F  E8 0008E 4$:     BLBS    @#^X0000FE02, 5$        ; 0547
                        0C 0000FE02             9F          02  E1 00095         BBC     #2, @#^X0000FE02, 6$    ;
                                                65      09      A6  9E 0009D 5$:     MOVAB   T_MENU, DS$A_AUTO_OR_MENU   ; 0550
                                                6A          02  88 000A1         BISB2   #2, DS$GL_CRD_FLAGS         ; 0551
                                                6A          04  8A 000A4         BICB2   #4, DS$GL_CRD_FLAGS         ; 0552
                                                            11  11 000A7         BRB     7$                          ; 0547
                                                65      08      A6  9E 000A9 6$:     MOVAB   T_NULL, DS$A_AUTO_OR_MENU   ; 0556
                                                            01  DD 000AD         PUSHL   #T                          ; 0557
                                                    0000FE00    9F  DD 000AF         PUSHL   @#^X0000FE00                ;
                                                            7E  D4 000B5         CLRL    -(SP)                       ;
                                                68          03  FB 000B7         CALLS   #3, CRD_ERROR               ;
                                                50 0000FE17     9F  9A 000BA 7$:     MOVZBL  @#^X0000FE17, R0        ; 0564
                                                03              50  91 000C1         CMPB    R0, #3                  ;
                                                                09  13 000C4         BEQL    8$                      ;
                                                                01  DD 000C6         PUSHL   #1                      ; 0565
                                                                50  DD 000C8         PUSHL   R0                      ;
                                                                01  DD 000CA         PUSHL   #1                      ;
                                                68              03  FB 000CC         CALLS   #3, CRD_ERROR           ;
                                                        E4      A5  B5 000CF 8$:     TSTW    DS$Q_CRD_VECTOR_DESC    ; 0567
                                                                1C  12 000D2         BNEQ    11$                     ;
                                            E4  A5          64  8F 9B 000D4         MOVZBW  #100, DS$Q_CRD_VECTOR_DESC ; 0578
                                            E8  A5          80  A5 9E 000D9         MOVAB   DS$A_CRD_VECTORS, DS$Q_CRD_VECTOR_DESC+4 ; 0579
                                                51              69 98 000DE         CVTBL   DS$AL_DS_DISPATCH_VECTORS, R1 ; 0586
                                                50              01 CE 000E1         MNEGL   #1, VECTOR_INDEX        ; 0592
```

```
                                      06   11 000E4        BRB      10$                                            ;
              E8 B540               6940   D0 000E6  9$:   MOVL     DS$AL_DS_DISPATCH_VECTORS[VECTOR_INDEX], -     ; 0594
                                                                    @DS$Q_CRD_VECTOR_DESC+4[VECTOR_INDEX]
      F6          50                  51   F2 000EC 10$:   AOBLSS   R1, VECTOR_INDEX, 9$                          ; 0586
                                      01   DD 000F0 11$:   PUSHL    #1                                             ; 0612
                            04        AE   9F 000F2        PUSHAB   CRD_FILE_ATTRIBUTES                            ;
                            0C        AE   9F 000F5        PUSHAB   CRD_ACTUAL_LENGTH                              ;
                                      7E   7C 000F8        CLRQ     -(SP)                                          ;
                                      56   DD 000FA        PUSHL    R6                                             ;
                            F8        A6   9F 000FC        PUSHAB   T_CRD_IMAGE_NAME                               ;
                  6B                  07   FB 000FF        CALLS    #7, DS$LOAD                                    ;
                  52                  50   D0 00102        MOVL     R0, STATUS                                     ;
                  09                  52   E8 00105        BLBS     STATUS, 12$                                    ;
                                      01   DD 00108        PUSHL    #1                                             ; 0614
                                      52   DD 0010A        PUSHL    STATUS                                         ;
                                      02   DD 0010C        PUSHL    #2                                             ;
                  68                  03   FB 0010E        CALLS    #3, CRD_ERROR                                  ;
            F4    A5  00000000G       EF   D0 00111 12$:   MOVL     DS$GA_LASTADR, DS$A_SAVED_LASTADR              ; 0616
      50    04    AE  000001FF        8F   C1 00119        ADDL3    #511, CRD_ACTUAL_LENGTH, R0                    ; 0619
                  50  00000200        8F   C6 00122        DIVL2    #512, PAGE_COUNT                               ;
      10 0000FE03 9F                  04   E1 00129        BBC      #4, @#^X0000FE03, 13$                          ; 0622
                                      7E   7C 00131        CLRQ     -(SP)                                          ; 0632
                            10        AE   9F 00133        PUSHAB   CRD_IMAGE_LOCATION                             ;
                                      50   DD 00136        PUSHL    PAGE_COUNT                                     ;
         00000000G  00                04   FB 00138        CALLS    #4, SYS$EXPREG                                 ;
                                      10   11 0013F        BRB      14$                                            ;
                                      03   DD 00141 13$:   PUSHL    #3                                             ; 0642
                                      7E   D4 00143        CLRL     -(SP)                                          ;
                            10        AE   9F 00145        PUSHAB   CRD_IMAGE_LOCATION                             ;
                                      50   DD 00148        PUSHL    PAGE_COUNT                                     ; 0643
         00000000G  EF                04   FB 0014A        CALLS    #4, CRD$EXPREG                                 ;
                    52                50   D0 00151 14$:   MOVL     R0, STATUS                                     ;
                    53      08        AE   D0 00154        MOVL     CRD_IMAGE_LOCATION, R3                         ; 0653
      50          0C  AE              53   C3 00158        SUBL3    R3, CRD_IMAGE_LOCATION+4, R0                   ;
                  EC  A5    01        A0   9E 0015D        MOVAB    1(R0), DS$Q_CRD_IMAGE_DESC                     ;
                  F0  A5              53   D0 00162        MOVL     R3, DS$Q_CRD_IMAGE_DESC+4                      ; 0655
                      23              52   E8 00166        BLBS     STATUS, 17$                                    ; 0657
         FFFFFFFF     8F              53   D1 00169        CMPL     R3, #-1                                        ; 0662
                      11              11   12 00170        BNEQ     15$                                            ;
         FFFFFFFF     8F    0C        AE   D1 00172        CMPL     CRD_IMAGE_LOCATION+4, #-1                      ; 0663
                                      07   12 0017A        BNEQ     15$                                            ;
                            EC        A5   7C 0017C        CLRQ     DS$Q_CRD_IMAGE_DESC                            ; 0666
                                      01   DD 0017F        PUSHL    #1                                             ; 0669
                                      02   11 00181        BRB      16$                                            ;
                                      7E   D4 00183 15$:   CLRL     -(SP)                                          ; 0672
                                      52   DD 00185 16$:   PUSHL    STATUS                                         ;
                                      03   DD 00187        PUSHL    #3                                             ;
                  68                  03   FB 00189        CALLS    #3, CRD_ERROR                                  ;
                                      01   DD 0018C 17$:   PUSHL    #1                                             ; 0682
                            04        AE   9F 0018E        PUSHAB   CRD_FILE_ATTRIBUTES                            ;
                            0C        AE   9F 00191        PUSHAB   CRD_ACTUAL_LENGTH                              ;
                                      53   DD 00194        PUSHL    R3                                             ;
                            14        AE   DD 00196        PUSHL    CRD_ACTUAL_LENGTH                              ;
                                      56   DD 00199        PUSHL    R6                                             ;
                            F8        A6   9F 0019B        PUSHAB   T_CRD_IMAGE_NAME                               ;
                  6B                  07   FB 0019E        CALLS    #7, DS$LOAD                                    ;
                  52                  50   D0 001A1        MOVL     R0, STATUS                                     ;
```

```
                                09              52  E8 001A4                BLBS    STATUS, 18$                                     ; 0684
                                                7E  D4 001A7                CLRL    -(SP)                                           ; 0685
                                                52  DD 001A9                PUSHL   STATUS                                          ;
                                                04  DD 001AB                PUSHL   #4                                              ;
                                68              03  FB 001AD                CALLS   #3, CRD_ERROR                                   ;
                     00000000G  EF        04    AE  C0 001B0 18$:           ADDL2   CRD_ACTUAL_LENGTH, DS$GA_LASTADR                ; 0687
                                66              54  E9 001B8                BLBC    R4, 19$                                         ; 0690
                                          F0    A5  DD 001BB                PUSHL   DS$Q_CRD_IMAGE_DESC+4                           ; 0698
                                        0557    C6  9F 001BE                PUSHAB  T_TRACE_1                                       ;
                                                01  DD 001C2                PUSHL   #1                                              ;
                                                1A  DD 001C4                PUSHL   #26                                             ;
                                67              04  FB 001C6                CALLS   #4, DSX$PRINT                                   ;
                                          EC    A5  DD 001C9                PUSHL   DS$Q_CRD_IMAGE_DESC                             ; 0699
                                        058C    C6  9F 001CC                PUSHAB  T_TRACE_2                                       ;
                                                01  DD 001D0                PUSHL   #1                                              ;
                                                1A  DD 001D2                PUSHL   #26                                             ;
                                67              04  FB 001D4                CALLS   #4, DSX$PRINT                                   ;
                                7E              63  98 001D7                CVTBL   (R3), -(SP)                                     ; 0703
                                7E              69  98 001DA                CVTBL   DS$AL_DS_DISPATCH_VECTORS, -(SP)                ;
                                        05C3    C6  9F 001DD                PUSHAB  T_TRACE_4                                       ;
                                                01  DD 001E1                PUSHL   #1                                              ;
                                                1A  DD 001E3                PUSHL   #26                                             ;
                                67              05  FB 001E5                CALLS   #5, DSX$PRINT                                   ;
                                7E        01    A3  98 001E8                CVTBL   1(R3), -(SP)                                    ; 0707
                                7E        01    A9  98 001EC                CVTBL   DS$AL_DS_DISPATCH_VECTORS+1, -(SP)              ;
                                        0606    C6  9F 001F0                PUSHAB  T_TRACE_5                                       ;
                                                01  DD 001F4                PUSHL   #1                                              ;
                                                1A  DD 001F6                PUSHL   #26                                             ;
                                67              05  FB 001F8                CALLS   #5, DSX$PRINT                                   ;
                                7E        02    A3  98 001FB                CVTBL   2(R3), -(SP)                                    ; 0711
                                7E        02    A9  98 001FF                CVTBL   DS$AL_DS_DISPATCH_VECTORS+2, -(SP)              ;
                                        0649    C6  9F 00203                PUSHAB  T_TRACE_6                                       ;
                                                01  DD 00207                PUSHL   #1                                              ;
                                                1A  DD 00209                PUSHL   #26                                             ;
                                67              05  FB 0020B                CALLS   #5, DSX$PRINT                                   ;
                                7E        03    A3  98 0020E                CVTBL   3(R3), -(SP)                                    ; 0715
                                7E        03    A9  98 00212                CVTBL   DS$AL_DS_DISPATCH_VECTORS+3, -(SP)              ;
                                        068C    C6  9F 00216                PUSHAB  T_TRACE_7                                       ;
                                                01  DD 0021A                PUSHL   #1                                              ;
                                                1A  DD 0021C                PUSHL   #26                                             ;
                                67              05  FB 0021E                CALLS   #5, DSX$PRINT                                   ;
                                                53  DD 00221 19$:           PUSHL   R3                                              ; 0718
                     00000000V  EF              01  FB 00223                CALLS   #1, EXCHANGE_DISPATCH_VECTORS                   ;
                                                04  0022A                   RET                                                     ; 0722

; Routine Size:  555 bytes,    Routine Base:  CODE + 0000

;   0723  1     %SBTTL 'DSR$Unload_CRD Routine'
;   0724  1
;   0725  1     GLOBAL ROUTINE DSR$Unload_CRD (Return_To_Caller) : NOVALUE =
;   0726  1
;   0727  1     !++
;   0728  1     ! Functional Description:
;   0729  1     !
;   0730  1     !       'Unload' the CRD image from memory. This routines does not actually unload the CRD code and data per se,
;   0731  1     !       but it does deallocate the memory that the CRD code and data occupies. This routine also deallocates
```

```
;  0732  1        !       all temporary CRD storage that was allocated in the DSR$Load_CRD routine.
;  0733  1        !
;  0734  1        ! Formal Parameters:
;  0735  1        !
;  0736  1        !       Return_To_Caller: If the image is loaded, then return to the caller of this routine when the
;  0737  1        !                         unloading is done. If it is not loaded, return regardless.
;  0738  1        !
;  0739  1        ! Side Effects:
;  0740  1        !
;  0741  1        !       Too numerous to mention. See the code!
;  0742  1        !
;  0743  1        ! Completion Codes:
;  0744  1        !
;  0745  1        !       None
;  0746  1        !--
;  0747  2        BEGIN                                   ! Routine DSR$Unload_CRD
;  0748  2                BIND
;  0749  2                        CRD_Vectors = (.DS$Q_CRD_Image_Desc [K_Image_Ptr]) : VECTOR [, BYTE, SIGNED];
;  0750  2
;  0751  2                REGISTER
;  0752  2                        Temp_Vector_Storage_Length;
;  0753  2
;  0754  2                MAP
;  0755  2                        DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE, SIGNED];
;  0756  2
;  0757  2                Module_Debug ('DSR$Unload_CRD - Start:');                                       !E07
;  0758  2
;  0759  2                !+
;  0760  2                ! Clear out the three ^C handler addresses. This must be done before going back to the CLI.
;  0761  2                !-
;  0762  2
;  0763  2                DS$GA_DS_Ctrl_C_First  = 0;
;  0764  2                DS$GA_DS_Ctrl_C_Second = 0;
;  0765  2                DS$L_UserCntrlC        = 0;
;  0766  2
;  0767  2                $DS_CntrlC (Disabl = 1);        ! Disable ^C recognition
;  0768  2
;  0769  2                !+
;  0770  2                ! Check first to see if the CRD image is in there. This is checked by looking at the
;  0771  2                ! image descriptor used to point to the image. If this is null, then nothing has been
;  0772  2                ! loaded in yet. Note that the Vector_Desc has been set up already, but the vectors
;  0773  2                ! have not been exchanged yet. So, it doesn't matter that the Vector_Desc isn't null.
;  0774  2                !-
;  0775  2
;  0776  2                IF (.DS$Q_CRD_Image_Desc [K_Image_Size] EQL 0) THEN
;  0777  3                        BEGIN
;  0778  3                        Module_Debug ('DSR$Unload_CRD - Short end:');                           !E07
;  0779  3
;  0780  3                        $DS_CntrlC (Disabl = 0);        ! Enable ^C recognition
;  0781  3                        RETURN;
;  0782  2                        END;
;  0783  2
;  0784  2                !+
;  0785  2                ! If the Saved_LastAdr is not zero, restore the LastAdr address for the VDS.
;  0786  2                !-
;  0787  2
;  0788  2                IF (.DS$A_Saved_LastAdr NEQ 0) THEN
```

```
;    0789  3                    BEGIN
;    0790  3                    DS$GA_LastAdr = .DS$A_Saved_LastAdr;
;    0791  3                    DS$A_Saved_LastAdr = 0;
;    0792  2                    END;
;    0793  2
;    0794  2                !+
;    0795  2                ! Now check to see if an internal error occurred while running CRD. If an internal error did happen,
;    0796  2                ! then the Positive_Version will be one less than the absolute value of the Negative_Version - the
;    0797  2                ! CRD_Internal_Error routine does this.
;    0798  2                !-
;    0799  2
;    0800  3                IF ((.CRD_Vectors [K_Positive_Version] - 1) EQL
;    0801  3                   (- .CRD_Vectors [K_Negative_Version]))
;    0802  2                THEN
;    0803  2                        !+
;    0804  2                        ! Simply call the error routine so that it can print some error messages, and return to here.
;    0805  2                        !-
;    0806  2                        CRD_Error (K_CRD_Internal, 0, 0);
;    0807  2
;    0808  2                !+
;    0809  2                ! Now transfer the dispatch vectors back from the temporary vector storage to where they
;    0810  2                ! belong - in the DS$AL_DS_Dispatch_Vectors area. This enables CRD to be started again.
;    0811  2                !-
;    0812  2
;    0813  2                Temp_Vector_Storage_Length = .DS$Q_CRD_Vector_Desc [DSC$W_Length];
;    0814  2
;    0815  2                IF (.Temp_Vector_Storage_Length NEQ 0) THEN
;    0816  3                        BEGIN
;    0817  3                        REGISTER
;    0818  3                                Number_Of_Vectors;
;    0819  3
;    0820  4                                BEGIN
;    0821  4                                BIND
;    0822  4                                        Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, BYTE];
;    0823  4
;    0824  4                                Number_Of_Vectors = .Temp_Vectors [K_Number_Of_Vectors];
;    0825  3                                END;
;    0826  3
;    0827  3                        INCR Vector_Index FROM 0 TO .Number_Of_Vectors DO
;    0828  4                                BEGIN
;    0829  4                                MAP
;    0830  4                                        DS$AL_DS_Dispatch_Vectors : VECTOR [, LONG];
;    0831  4
;    0832  4                                BIND
;    0833  4                                        Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, LONG];
;    0834  4
;    0835  4                                DS$AL_DS_Dispatch_Vectors [.Vector_Index] = .Temp_Vectors [.Vector_Index];
;    0836  3                                END;
;    0837  2                        END;                    ! IF . . .
;    0838  2
;    0839  2                !+
;    0840  2                ! Now contract the region occupied by the CRD image.
;    0841  2                !-
;    0842  2
;    0843  2                IF (.DS$Q_CRD_Image_Desc [K_Image_Size] NEQ 0) THEN
;    0844  3                        BEGIN
;    0845  3                        REGISTER
```

```
;    0846  3                        Status,
;    0847  3                        Page_Count;
;    0848  3
;    0849  3                LOCAL
;    0850  3                        Temp_Vector : VECTOR [2, LONG];
;    0851  3
;    0852  3                Page_Count = (.DS$Q_CRD_Image_Desc [K_Image_Size] + 511) / 512;
;    0853  3
;    0854  5                IF (NOT (
;    0855  5                        IF (.DSA$V_User) THEN
;  P 0856  5                                Status = $CntReg (
;  P 0857  5                                        PagCnt = .Page_Count,
;  P 0858  5                                        RetAdr = Temp_Vector,
;    0859  6                                        Region = 0)
;    0860  5                        ELSE
;  P 0861  5                                Status = $CntReg (
;  P 0862  5                                        PagCnt = .Page_Count,
;  P 0863  5                                        RetAdr = Temp_Vector,
;    0864  5                                        Region = 3);
;    0865  4                        .Status))
;    0866  3                THEN
;    0867  4                        BEGIN
;    0868  4                        CRD_Error (K_Not_Contracted, .Status, K_Region_Expanded);
;    0869  4                        END
;    0870  2                END;
;    0871  2
;    0872  2                !+
;    0873  2                ! Now, clear out the descriptors, in preparation for next time.
;    0874  2                !-
;    0875  2
;    0876  2                DS$Q_CRD_Image_Desc [K_Image_Size] = 0;
;    0877  2                DS$Q_CRD_Image_Desc [K_Image_Ptr]  = 0;
;    0878  2
;    0879  2                DS$Q_CRD_Vector_Desc [DSC$W_Length]  = 0;
;    0880  2                DS$Q_CRD_Vector_Desc [DSC$B_Class]   = 0;
;    0881  2                DS$Q_CRD_Vector_Desc [DSC$B_DType]   = 0;
;    0882  2                DS$Q_CRD_Vector_Desc [DSC$A_Pointer] = 0;
;    0883  2
;    0884  2                INCR Vec_Index FROM 0 TO (CRD$K_Numb_Dispatch_Vectors - 1) DO
;    0885  2                        DS$A_CRD_Vectors [.Vec_Index] = %X'FFFFFFFF';
;    0886  2
;    0887  2                Module_Debug ('DSR$Unload_CRD - Long end:');                    !E07
;    0888  2
;    0889  2                !+
;    0890  2                ! Now resume echoing ^C's.                                     !E06
;    0891  2                !-
;    0892  2
;    0893  2                CRD$V_Ctrl_C_No_Echo = false;                                  !E07
;    0894  2                CRD$V_Flush_Ctrl_C = false;                                    !E07
;    0895  2
;    0896  2                IF (.Return_To_Caller) THEN
;    0897  2                        RETURN                  ! Return to this routine's caller
;    0898  2                ELSE
;    0899  2                        CRD_Exit ();            ! Either go to Begin or Halt
;    0900  1    END;                                        ! Routine DSR$Unload_CRD
```

```
                                                                .EXTRN  DS$CNTRLC, SYS$CNTREG

                                                003C 00000      .ENTRY  DSR$UNLOAD_CRD, Save R2,R3,R4,R5                ; 0725
                                    55 00000000V EF  9E 00002   MOVAB   CRD_ERROR, R5                                   ;
                                    54 00C00000G 9F  9E 00009   MOVAB   @#DS$CNTRLC, R4                                 ;
                                    53 00000000' EF  9E 00010   MOVAB   DS$Q_CRD_IMAGE_DESC, R3                         ;
                                    5E           08  C2 00017   SUBL2   #8, SP                                        ;
                                    52       04  A3  D0 0001A   MOVL    DS$Q_CRD_IMAGE_DESC+4, R2                     ; 0749
                                       00000000G EF  D4 0001E   CLRL    DS$GA_DS_CTRL_C_FIRST                         ; 0763
                                       00000000G EF  D4 00024   CLRL    DS$GA_DS_CTRL_C_SECOND                        ; 0764
                                       00000000G EF  D4 0002A   CLRL    DS$L_USERCNTRLC                               ; 0765
                                                 01  DD 00030   PUSHL   #1                                            ; 0767
                                                 7E  D4 00032   CLRL    -(SP)                                         ;
                                    64           02  FB 00034   CALLS   #2, DS$CNTRLC                                 ;
                                                 63  D5 00037   TSTL    DS$Q_CRD_IMAGE_DESC                           ; 0776
                                                 06  12 00039   BNEQ    1$                                            ;
                                                 7E  7C 0003B   CLRQ    -(SP)                                         ; 0780
                                    64           02  FB 0003D   CALLS   #2, DS$CNTRLC                                 ;
                                                 04  00040   RET                                                       ; 0777
                                    50       08  A3  D0 00041 1$:   MOVL    DS$A_SAVED_LASTADR, R0                        ; 0788
                                                 0A  13 00045   BEQL    2$                                            ;
                          00000000G EF           50  D0 00047   MOVL    R0, DS$GA_LASTADR                             ; 0790
                                       08  A3  D4 0004E   CLRL    DS$A_SAVED_LASTADR                            ; 0791
                                    51     01  A2  98 00051 2$:   CVTBL   1(R2), R1                                    ; 0800
                                                 51  D7 00055   DECL    R1                                            ;
                                    50     02  A2  98 00057   CVTBL   2(R2), R0                                    ; 0801
                                    50           50  CE 0005B   MNEGL   R0, R0                                       ;
                                    50           51  D1 0005E   CMPL    R1, R0                                        ;
                                                 07  12 00061   BNEQ    3$                                            ;
                                                 7E  7C 00063   CLRQ    -(SP)                                         ; 0806
                                                 05  DD 00065   PUSHL   #5                                            ;
                                    65           03  FB 00067   CALLS   #3, CRD_ERROR                                 ;
                                    50     F8  A3  3C 0006A 3$:   MOVZWL  DS$Q_CRD_VECTOR_DESC, -                     ; 0813
                                                                   TEMP_VECTOR_STORAGE_LENGTH
                                                 17  13 0006E   BEQL    6$                                            ; 0815
                                    50     FC  B3  9A 00070   MOVZBL  @DS$Q_CRD_VECTOR_DESC+4, NUMBER_OF_VECTORS       ; 0824
                                    51           01  CE 00074   MNEGL   #1, VECTOR_INDEX                               ; 0827
                                                 0A  11 00077   BRB     5$                                            ;
                       00000000GEF41     FC B341  D0 00079 4$:   MOVL    @DS$Q_CRD_VECTOR_DESC+4[VECTOR_INDEX], -      ; 0835
                                                                   DS$AL_DS_DISPATCH_VECTORS[VECTOR_INDEX]
                F2                 51           50  F3 00083 5$:   AOBLEQ  NUMBER_OF_VECTORS, VECTOR_INDEX, 4$         ; 0827
                                    50           63  D0 00087 6$:   MOVL    DS$Q_CRD_IMAGE_DESC, R0                       ; 0843
                                                 34  13 0008A   BEQL    9$                                            ;
                                    50     01FF  C0  9E 0008C   MOVAB   511(R0), R0                                   ; 0852
                                    50 00000200  8F  C6 00091   DIVL2   #512, PAGE_COUNT                              ;
                04 0000FE03         9F           04  E1 00098   BBC     #4, @#^X0000FE03, 7$                          ; 0855
                                                 7E  D4 000A0   CLRL    -(SP)                                         ; 0859
                                                 02  11 000A2   BRB     8$                                            ;
                                                 03  DD 000A4 7$:   PUSHL   #3                                        ; 0864
                                                 7E  D4 000A6 8$:   CLRL    -(SP)                                     ;
                                       08  AE  9F  000A8   PUSHAB  TEMP_VECTOR                                       ;
                                                 50  DD 000AB   PUSHL   PAGE_COUNT                                    ;
                          00000000G  00          04  FB 000AD   CALLS   #4, SYS$CNTREG                                ;
                                                 09  50 E8 000B4   BLBS    STATUS, 9$                                  ; 0865
                                                 7E  D4 000B7   CLRL    -(SP)                                         ; 0868
                                                 50  DD 000B9   PUSHL   STATUS                                        ;
```

```
                                                08  DD 000BB          PUSHL   #8                                                      ;
                                          65    03  FB 000BD          CALLS   #3, CRD_ERROR                                           ;
                                                63  7C 000C0 9$:      CLRQ    DS$Q_CRD_IMAGE_DESC                                     ; 0876
                                             F8 A3  7C 000C2          CLRQ    DS$Q_CRD_VECTOR_DESC                                    ; 0879
                                                50  D4 000C5          CLRL    VEC_INDEX                                               ; 0884
                                       94 A340      01  CE 000C7 10$:  MNEGL   #1, DS$A_CRD_VECTORS[VEC_INDEX]                        ; 0885
                                F7        50        0D  F3 000CC          AOBLEQ  #13, VEC_INDEX, 10$                                     ;
                             00000000G    EF        06  8A 000D0          BICB2   #6, DS$GL_CRD_FLAGS                             ; 0894
                                          07     04 AC  E8 000D7          BLBS    RETURN_TO_CALLER, 11$                           ; 0896
                             00000000V    EF        00  FB 000DB          CALLS   #0, CRD_EXIT                                    ; 0899
                                                04  000E2 11$:      RET                                                             ; 0900

; Routine Size:  227 bytes,    Routine Base:  CODE + 0228

;   0901  1     %SBTTL 'DSR$Restore_CRD_Vectors Routine'
;   0902  1
;   0903  1     GLOBAL ROUTINE DSR$Restore_CRD_Vectors : NOVALUF =
;   0904  1
;   0905  1     !++
;   0906  1     ! Functional Description:
;   0907  1     !
;   0908  1     !       Using the vectors stored in the DS$Q_CRD_Vector_Desc, restore the CRD dispatch vectors. These vectors
;   0909  1     !       are originally set up the the DSVECS module. NOTE: This routine SHOULD ONLY be called from the BEGIN0
;   0910  1     !       label in the KERNEL module!!!!!!
;   0911  1     !
;   0912  1     ! Formal Parameters:
;   0913  1     !
;   0914  1     !       None
;   0915  1     !
;   0916  1     ! Side Effects:
;   0917  1     !
;   0918  1     !       Changes the CRD dispatch vectors located at DS$AL_DS_Dispatch_Vectors
;   0919  1     !
;   0920  1     ! Completion Codes:
;   0921  1     !
;   0922  1     !       None
;   0923  1     !--
;   0924  2     BEGIN                                       ! Routine DSR$Restore_CRD_Vectors
;   0925  2             Module_Debug ('DSR$Restore_CRD_Vectors - Start:');                                      !E07
;   0926  2
;   0927  2             !+
;   0928  2             ! Now resume echoing ^C's. Do this regardless of whether or not CRD is in there (just to be sure).      !E06
;   0929  2             !-
;   0930  2
;   0931  2             CRD$V_Ctrl_C_No_Echo = False;                                                           !E07
;   0932  2             CRD$V_Flush_Ctrl_C = False;                                                             !E07
;   0933  2
;   0934  2             !+
;   0935  2             ! Now transfer the dispatch vectors back from the vector storage area to where they
;   0936  2             ! belong - in the DS$AL_DS_Dispatch_Vectors area. This enables CRD to be started again.
;   0937  2             ! Do this ONLY if the vector were changed at some previous point. Determine this by
;   0938  2             ! checking the descriptor that points to the vector storage area for the vectors.
;   0939  2             !-
;   0940  2
;   0941  3             IF ((.DS$Q_CRD_Vector_Desc [DSC$W_Length] NEQ 0) AND
;   0942  3                 (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer] NEQ 0))
```

```
;    0943  2              THEN
;    0944  3                      BEGIN
;    0945  3                      !+
;    0946  3                      ! This will ONLY be done iff CRD was previously running and was stopped by a ^P, then a
;    0947  3                      ! S 10000 console command was done. If this sequence of events is done, then we must restore
;    0948  3                      ! the stored dispatch vectors to their proper place in the DS$AL_DS_Dispatch_Vectors area.
;    0949  3                      ! Also, since we could have been executing a diagnostic at the time the ^P was typed, we must
;    0950  3                      ! clear out the Environ longword in the diagnostic header so that the VDS does not think a
;    0951  3                      ! diagnostic is still loaded once the VDS comes up again (after the S 10000).
;    0952  3                      !-
;    0953  3
;    0954  3                      REGISTER
;    0955  3                              Number_Of_Vectors;
;    0956  3
;    0957  3                      BIND
;    0958  3                              Vector_Info = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, BYTE];
;    0959  3
;    0960  3                      Number_Of_Vectors = .Vector_Info [K_Number_Of_Vectors];
;    0961  3
;    0962  3                      INCR Vector_Index FROM 0 TO .Number_Of_Vectors DO
;    0963  4                              BEGIN
;    0964  4                              MAP
;    0965  4                                      DS$AL_DS_Dispatch_Vectors : VECTOR [, LONG];
;    0966  4
;    0967  4                              BIND
;    0968  4                                      Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, LONG];
;    0969  4
;    0970  4                              DS$AL_DS_Dispatch_Vectors [.Vector_Index] = .Temp_Vectors [.Vector_Index];
;    0971  3                              END;
;    0972  3
;    0973  3                      L$L_Environ = 0;
;    0974  3
;    0975  3                      DS$A_Saved_LastAdr = 0;
;    0976  3
;    0977  3                      DS$Q_CRD_Image_Desc [K_Image_Size] = 0;
;    0978  3                      DS$Q_CRD_Image_Desc [K_Image_Ptr]  = 0;
;    0979  3
;    0980  3                      DS$Q_CRD_Vector_Desc [DSC$W_Length]  = 0;
;    0981  3                      DS$Q_CRD_Vector_Desc [DSC$B_Class]   = 0;
;    0982  3                      DS$Q_CRD_Vector_Desc [DSC$B_DType]   = 0;
;    0983  3                      DS$Q_CRD_Vector_Desc [DSC$A_Pointer] = 0;
;    0984  3
;    0985  3                      !+
;    0986  3                      ! Also set the temporary vector storage area to all -1's. This will allow the VDS to
;    0987  3                      ! catch all 'buggy' references to an address in this area.
;    0988  3                      !-
;    0989  3
;    0990  3                      INCR Vec_Index FROM 0 TO (CRD$K_Total_Numb_Vectors - 1) DO                              !E07
;    0991  3                              DS$A_CRD_Vectors [.Vec_Index] = %X'FFFFFFFF';
;    0992  2                      END;
;    0993  2              Module_Debug ('DSR$Restore_CRD_Vectors - End:');                                                !E07
;    0994  2              RETURN;
;    0995  1      END;                                    ! Routine DSR$Restore_CRD_Vectors
```

```
                                                0004 00000          .ENTRY  DSR$RESTORE_CRD_VECTORS, Save R2                        ; 0903
                                    52 00000000'  EF  9E 00002      MOVAB   DS$Q_CRD_VECTOR_DESC+4, R2                              ;
                                00000000G   EF        06  8A 00009  BICB2   #6, DS$GL_CRD_FLAGS                                     ; 0932
                                              FC      A2  B5 00010  TSTW    DS$Q_CRD_VECTOR_DESC                                    ; 0941
                                                      35  13 00013  BEQL    4$                                                      ;
                                                      62  D5 00015  TSTL    DS$Q_CRD_VECTOR_DESC+4                                  ; 0942
                                                      31  13 00017  BEQL    4$                                                      ;
                                    50          00    B2  9A 00019  MOVZBL  @DS$Q_CRD_VECTOR_DESC+4, NUMBER_OF_VECTORS             ; 0960
                                    51                01  CE 0001D  MNEGL   #1, VECTOR_INDEX                                       ; 0962
                                                      0A  11 00020  BRB     2$                                                      ;
                                00000000GEF41         00 B241  D0 00022 1$:   MOVL    @DS$Q_CRD_VECTOR_DESC+4[VECTOR_INDEX], -        ; 0970
                                                                              DS$AL_DS_DISPATCH_VECTORS[VECTOR_INDEX]                 ;
                            F2              51                50  F3 0002C 2$:   AOBLEQ  NUMBER_OF_VECTORS, VECTOR_INDEX, 1$            ; 0962
                                                00000204      9F  D4 00030      CLRL    @#^X00000204                                    ; 0971
                                                      04      A2  D4 00036      CLRL    DS$Q_CRD_IMAGE_DESC                             ; 0971
                                                      08      A2  7C 00039      CLRQ    DS$Q_CRD_IMAGE_DESC+4                           ; 0970
                                                      FC      A2  7C 0003C      CLRQ    DS$Q_CRD_VECTOR_DESC                            ; 0980
                                                      50      D4 0003F          CLRL    VEC_INDEX                                       ; 0990
                                        98 A240               01  CE 00041 3$:   MNEGL   #1, DS$A_CRD_VECTORS[VEC_INDEX]           ; 0991
                            F7                  50            18  F3 00046      AOBLEQ  #24, VEC_INDEX, 3$                              ;
                                                          04  0004A 4$:   RET                                                         ; 0995

; Routine Size:  75 bytes,    Routine Base:  CODE + 030E

;  0996  1      %SBTTL 'Exchange_Dispatch_Vectors Routine'
;  0997  1
;  0998  1      GLOBAL ROUTINE Exchange_Dispatch_Vectors (A_CRD_Dispatch_Vectors) : NOVALUE =                                    !C11
;  0999  1
;  1000  1      !++
;  1001  1      ! Functional Description:
;  1002  1      !
;  1003  1      !       Exchange the CRD and DS dispatch vectors. That is, the DS has a set of dispatch vectors that CRD uses to
;  1004  1      !       access certain DS locations (such as the DSA flags, the Control-C handler address, etc.). These are
;  1005  1      !       stored in the CRDVECS module (part of the Resident-DS image). CRD also has a set of dispatch vectors
;  1006  1      !       that the Resident-DS will use to get to certain CRD locations and routines (such as the CRD routine to
;  1007  1      !       call when something is being printed by the DS). These are stored at the beginning of the CRD-Loadable
;  1008  1      !       image (i.e., in the very first Psect). Exchange_Dispatch_Vectors will exchange these two sets of vectors,
;  1009  1      !       thus enabling the DS to get to the CRD routines, and enabling CRD to get to the DS locations and routines.
;  1010  1      !
;  1011  1      !       Before:
;  1012  1      !               CRD_Dispatch_Vectors:           Contains the addresses of the CRD-Loadable routines and locations
;  1013  1      !               DS$AL_DS_Dispatch_Vectors:      Contains the addresses of certain DS locations and routines
;  1014  1      !       After:
;  1015  1      !               CRD_Dispatch_Vectors:           Contains the addresses of certain DS locations and routines
;  1016  1      !               DS$AL_DS_Dispatch_Vectors:      Contains the addresses of the CRD-Loadable routines and locations
;  1017  1      !
;  1018  1      !       Note also that the first longword in each set of vectors will contain a Count byte (the number of
;  1019  1      !       dispatch vectors), a null byte, and two bytes containing the version number of CRD. One of the version
;  1020  1      !       numbers will be positive, and the other one will be negative. The algorithm for verification is as follows:
;  1021  1      !
;  1022  1      !               1. Read in the CRD-Loadable file
;  1023  1      !               2. Set CRD_Dispatch_Vectors = starting address of the now resident CRD image
;  1024  1      !               3. DS$AL_DS_Dispatch_Vectors [0] <24, 8, 0> = 0                         ! The null byte
;  1025  1      !               4. DS$AL_DS_Dispatch_Vectors [0] <16, 8, 0> =                           ! The version #'s
;  1026  1      !                   - DS$AL_DS_Dispatch_Vectors [0] <8, 8, 0>
```

```
;    1027  1        !               5. CRD_Dispatch_Vectors [0] <24, 8, 0> = 0                                      ! The null byte
;    1028  1        !               6. CRD_Dispatch_Vectors [0] <16, 8, 0> = - CRD_Dispatch_Vectors [0] <8, 8, 0>   ! The version #'s
;    1029  1        !
;    1030  1        !       If any of the above statements are not true, an appropriate error message is printed and CRD-AUTOTEST
;    1031  1        !       is aborted. This ensures that not only do the version numbers agree, but also that the image read in is
;    1032  1        !       indeed the CRD-Loadable image file.
;    1033  1        !
;    1034  1        !       Note: The CRD-Loadable image will be linked with a /System=%X0000 qualifer, thus making all addresses relati
;    1035  1        !       to zero. The only address that this affects are those in the CRD_Dispatch_Vectors, the ones that are moved t
;    1036  1        !       the Resident-DS area. These CRD routine addresses must be modified by the amount of the starting location of
;    1037  1        !       the CRD image (i.e., the parameter to this routine).
;    1038  1        ! Picture:
;    1039  1        !
;    1040  1        !                               C R D - L O A D A B L E
;    1041  1        !                               -----------------------
;    1042  1        !                       24              16              8       0 - Bit numbers
;    1043  1        !                       ----------------------------------------
;    1044  1        !                       |   0   | - Version # | + Version # |  Count  |  :  .CRD_Dispatch_Vectors [0]
;    1045  1        !                       |-------|-------------|-------------|---------|
;    1046  1        !                       |               CRD_Dispatch_Vectors [1]        |
;    1047  1        !                       |-----------------------------------------------|
;    1048  1        !                       |               CRD_Dispatch_Vectors [2]        |
;    1049  1        !                       |-----------------------------------------------|
;    1050  1        !                       |                       ...                     |
;    1051  1        !                       |-----------------------------------------------|
;    1052  1        !                       |             CRD_Dispatch_Vectors [.Count]     |
;    1053  1        !                       |-----------------------------------------------|
;    1054  1        !
;    1055  1        !
;    1056  1        !                               D S - R E S I D E N T
;    1057  1        !                               ---------------------
;    1058  1        !                       24              16              8       0 - Bit numbers
;    1059  1        !                       ----------------------------------------
;    1060  1        !                       |   0   | - Version # | + Version # |  Count  |  :  .DS$AL_DS_Dispatch_Vectors [0]
;    1061  1        !                       |-------|-------------|-------------|---------|
;    1062  1        !                       |               DS_Vectors [1]                  |
;    1063  1        !                       |-----------------------------------------------|
;    1064  1        !                       |               DS_Vectors [2]                  |
;    1065  1        !                       |-----------------------------------------------|
;    1066  1        !                       |                       ...                     |
;    1067  1        !                       |-----------------------------------------------|
;    1068  1        !                       |             DS_Vectors [.Count]               |
;    1069  1        !                       |-----------------------------------------------|
;    1070  1        !
;    1071  1        ! Formal Parameters:
;    1072  1        !
;    1073  1        !       A_CRD_Dispatch_Vectors :    The address of the CRD dispatch vectors. These vectors will contain the addresses
;    1074  1        !                                   of the DS routines and locations that CRD will have to access.
;    1075  1        ! Implicit Inputs:
;    1076  1        !
;    1077  1        !       DS$AL_DS_Dispatch_Vectors : The address of the DS dispatch vectors. These vectors will contain the addresses
;    1078  1        !                                   of the CRD routines and locations that the Resident-DS will have to access.
;    1079  1        !
;    1080  1        ! Side Effects:
;    1081  1        !
;    1082  1        !       Exchanges the DS and the CRD dispatch vectors.
;    1083  1        !
```

```
;    1084  1      ! Completion Codes:
;    1085  1      !
;    1086  1      !       None - If the verification process fails, the CRD_Error routine is called.
;    1087  1      !--
;    1088  2      BEGIN                                   ! Routine Exchange_Dispatch_Vectors
;    1089  2              MACRO
;  M 1090  2                      Corrupted_Error =
;  M 1091  2                              CRD_Error (K_Corrupted, 0, K_Region_Expanded)
;    1092  2                      %,
;    1093  2
;  M 1094  2                      Incompatibility_Error =
;  M 1095  2                              CRD_Error (K_Incompatible, 0, K_Region_Expanded)
;    1096  2                      %;
;    1097  2
;    1098  2              !+
;    1099  2              ! Use these when referencing the vectors (the longwords).
;    1100  2              !-
;    1101  2
;    1102  2              BIND
;    1103  2                      CRD_Dispatch_Vectors = (.A_CRD_Dispatch_Vectors)   : VECTOR [, LONG],
;    1104  2                      DS_Dispatch_Vectors  = (DS$AL_DS_Dispatch_Vectors) : VECTOR [, LONG];
;    1105  2
;    1106  2              !+
;    1107  2              ! Use these two when referencing the particular bytes in the first longword.
;    1108  2              !-
;    1109  2
;    1110  2              MAP
;    1111  2                      DS$AL_DS_Dispatch_Vectors :     VECTOR [, BYTE, SIGNED],
;    1112  2                      A_CRD_Dispatch_Vectors    : REF VECTOR [, BYTE, SIGNED];
;    1113  2
;    1114  2              !+
;    1115  2              ! Verify that the image file read in is indeed the CRD-Loadable image file.
;    1116  2              !-
;    1117  2
;    1118  2              IF (.A_CRD_Dispatch_Vectors [K_Null_Byte] NEQ 0) THEN
;    1119  2                      Corrupted_Error;
;    1120  2
;    1121  2              IF (.A_CRD_Dispatch_Vectors [K_Positive_Version] NEQ (- .A_CRD_Dispatch_Vectors [K_Negative_Version])) THEN
;    1122  2                      Corrupted_Error;
;    1123  2
;    1124  2              !+
;    1125  2              ! The format of the verification bytes is okay. Now check to see if the version numbers match
;    1126  2              ! (note: the DS version number must be either equal to or greater than the CRD version number).
;    1127  2              ! Also check to see if the vector counts are equal (see explanation on #2 in history above).          !C02
;    1128  2              !-
;    1129  2
;    1130  2              IF (.DS$AL_DS_Dispatch_Vectors [K_Positive_Version] LSS .A_CRD_Dispatch_Vectors [K_Positive_Version]) THEN
;    1131  2                      Incompatibility_Error;
;    1132  2
;    1133  2              IF (.DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors] NEQ .A_CRD_Dispatch_Vectors [K_Number_Of_Vectors]) THEN
;    1134  2                      Incompatibility_Error;                                                                        !C02
;    1135  2
;    1136  2              !+
;    1137  2              ! Exchange the two sets of vectors. Be sure and add in the CRD offset from zero. Since the CRD image
;    1138  2              ! is LINKed with a base of 0, all the .ADDRESSes in CRDVECS are based on a base of 0, rather than the
;    1139  2              ! actual base that is computed at run-time by the Load_CRD routine. So, I must add the value of the
;    1140  2              ! FIRST longword address in the CRD image to the values contained in each of the CRD dispatch vectors,
```

```
;    1141  2          ! and then store this value in the DS dispatch vector area.
;    1142  2          !-
;    1143  2
;    1144  2          INCR Vector FROM 1 TO (.DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors]) DO
;    1145  3                  BEGIN
;    1146  3                  REGISTER
;    1147  3                          Exchange;
;    1148  3
;    1149  3                  Exchange = .DS_Dispatch_Vectors [.Vector];
;    1150  3                  DS_Dispatch_Vectors [.Vector] = .CRD_Dispatch_Vectors [.Vector] + CRD_Dispatch_Vectors;
;    1151  3                                  ! Add in offset of CRD from zero
;    1152  3                  CRD_Dispatch_Vectors [.Vector] = .Exchange;
;    1153  2                  END;
;    1154  1      END;                                ! End of Routine Exchange_Dispatch_Vectors



                                                 003C 00000         .ENTRY  EXCHANGE_DISPATCH_VECTORS, Save R2,R3,R4,R5 ; 0998
                                55 00000000V     EF   9E 00002      MOVAB   CRD_ERROR, R5                               ;
                                54 00000000G     EF   9E 00009      MOVAB   DS$AL_DS_DISPATCH_VECTORS, R4               ;
                                52         04    AC   D0 00010      MOVL    A_CRD_DISPATCH_VECTORS, R2                  ; 1103
                                           03    A2   95 00014      TSTB    3(R2)                                       ; 1118
                                                 07   13 00017      BEQL    1$                                          ;
                                                 7E   7C 00019      CLRQ    -(SP)                                       ;
                                                 06   DD 0001B      PUSHL   #6                                          ;
                                             65       03   FB 0001D      CALLS   #3, CRD_ERROR                           ;
                                             50   02  A2   98 00020 1$:  CVTBL   2(R2), R0                             ; 1121
                                             50       50   CE 00024      MNEGL   R0, R0
                        50         01    A2           08   00   EC 00027      CMPV    #0, #8, 1(R2), R0
                                                      07   13 0002D      BEQL    2$
                                                      7E   7C 0002F      CLRQ    -(SP)
                                                      06   DD 00031      PUSHL   #6
                                             65       03   FB 00033      CALLS   #3, CRD_ERROR
                                         01  A2   01  A4   91 00036 2$:  CMPB    DS$AL_DS_DISPATCH_VECTORS+1, 1(R2)      ; 1130
                                                      07   18 0003B      BGEQ    3$
                                                      7E   7C 0003D      CLRQ    -(SP)
                                                      07   DD 0003F      PUSHL   #7
                                             65       03   FB 00041      CALLS   #3, CRD_ERROR
                                             62       64   91 00044 3$:  CMPB    DS$AL_DS_DISPATCH_VECTORS, (R2)         ; 1133
                                                      07   13 00047      BEQL    4$
                                                      7E   7C 00049      CLRQ    -(SP)
                                                      07   DD 0004B      PUSHL   #7
                                             65       03   FB 0004D      CALLS   #3, CRD_ERROR
                                             53       64   98 00050 4$:  CVTBL   DS$AL_DS_DISPATCH_VECTORS, R3          ; 1144
                                                      51   D4 00053      CLRL    VECTOR
                                                      0E   11 00055      BRB     6$
                                             50       6441 D0 00057 5$:  MOVL    DS_DISPATCH_VECTORS[VECTOR], EXCHANGE   ; 1149
                        6441             6241         52   C1 0005B      ADDL3   R2, (R2)[VECTOR], DS_DISPATCH_VECTORS-  ; 1150
                                                                                  [VECTOR]
                                         6241         50   D0 00061      MOVL    EXCHANGE, (R2)[VECTOR]                  ; 1152
                                EE       51           53   F3 00065 6$:  AOBLEQ  R3, VECTOR, 5$                         ; 1144
                                                           04   00069      RET                                         ; 1154

; Routine Size:  106 bytes,    Routine Base:  CODE + 0359
```

```
;    1155  1     %SBTTL 'CRD_Exit Routine'
;    1156  1
;    1157  1     ROUTINE CRD_Exit : NOVALUE =
;    1158  1
;    1159  1     !++
;    1160  1     ! Functional Description:
;    1161  1     !
;    1162  1     !       Exit from the CRD-world to either the VDS CLI command mode, or to the 11/730,725 console mode.
;    1163  1     !
;    1164  1     ! Formal Parameters:
;    1165  1     !
;    1166  1     !       None
;    1167  1     !
;    1168  1     ! Implicit Inputs:
;    1169  1     !
;    1170  1     !       None
;    1171  1     !
;    1172  1     ! Side Effects:
;    1173  1     !
;    1174  1     !       Either exits the VDS or goes to CLI mode via the Begin_Bliss label.
;    1175  1     !
;    1176  1     ! Completion Codes:
;    1177  1     !
;    1178  1     !       None - does not return from here
;    1179  1     !--
;    1180  2     BEGIN                                   ! Routine CRD_Exit
;    1181  2             BUILTIN
;    1182  2                     HALT;
;    1183  2
;    1184  2             Module_Debug ('CRD_Exit - Start:');                                             !C07
;    1185  2
;    1186  2             !+
;    1187  2             ! Cancel any ^C's that happened before this routine was called, and re-enable catching
;    1188  2             ! of ^C's.
;    1189  2             !-
;    1190  2
;    1191  2             DS$GL_Flags <DS$V_CtrlC> = Off;
;    1192  2             $DS_CntrlC (Disabl = 0);
;    1193  2
;    1194  2             !+
;    1195  2             ! Now resume echoing ^C's.                                                      !C06
;    1196  2             !-
;    1197  2
;    1198  2             CRD$V_Ctrl_C_No_Echo = False;                                                   !C07
;    1199  2             CRD$V_Flush_Ctrl_C = False;                                                     !C07
;    1200  2
;    1201  2             !+
;    1202  2             ! Restore the DSA and DS flag settings that were saved in DSR$Load_CRD.
;    1203  2             !-
;    1204  2
;    1205  2             DSA$GL_Flags = .DS$L_Saved_DSA_Flags;
;    1206  2             DS$GL_Flags  = .DS$L_Saved_DS_Flags;
;    1207  2
;    1208  2             !+
;    1209  2             ! Make sure all the CRD bits are cleared. Don't touch BINARY though. Perhaps the operator turned
;    1210  2             ! it on on purpose.
```

```
;    1211  2          !-
;    1212  2
;    1213  2          DSA$V_CRD_Trace     = Off;
;    1214  2          DSA$V_CRD_MenuTest_Off = Off;
;    1215  2                                    !                                    [10]
;    1216  2          DSA$V_CRD_MenuTest_On = Off;
;    1217  2                                    !                                    [10]
;    1218  2          DSA$V_CRD_AutoTest_Off = Off;
;    1219  2                                    !                                    [10]
;    1220  2
;    1221  2          DS_Cleanup ();            ! Cleanup any residual diagnostics
;    1222  2
;    1223  2          !+
;    1224  2          ! Make the VDS think that no diagnogstic is loaded.
;    1225  2          !-
;    1226  2
;    1227  2          DS$GL_Flags <DS$V_LodFlg> = Off;
;    1228  2          L$L_Environ = 0;
;    1229  2
;    1230  2          Module_Debug ('CRD_Exit - End:');                                !C07
;    1231  2
;    1232  2          IF (DSR$Check_MenuTest_Off_Set () OR
;    1233  2              DSR$Check_AutoTest_Off_Set ()
;    1234  2          ) THEN
;    1235  3                  BEGIN
;    1236  3                  WHILE (True) DO
;    1237  3                          HALT ();
;    1238  3                  END
;    1239  2          ELSE
;    1240  3                  BEGIN
;    1241  3                  !+
;    1242  3                  ! Since this goes back to the DS$Cli routine, the ^C recognition will be re-enabled there.
;    1243  3                  !-
;    1244  3
;    1245  3                  JSB_None (Begin_Bliss);
;    1246  3                  END;
;    1247  1      END;                                ! Routine CRD_Exit
```

```
                                        OFFC 00000 CRD_EXIT:
                                                   .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1157
                          52 00000000G  EF  9E 00002       MOVAB    DS$GL_FLAGS, R2                       ;
                          62            01  8A 00009       BICB2    #1, DS$GL_FLAGS                       ; 1191
                                        7E  7C 0000C       CLRQ     -(SP)                                 ; 1192
                00000000G 9F            02  FB 0000E       CALLS    #2, @#DS$CNTRLC                       ;
                00000000G EF            06  8A 00015       BICB2    #6, DS$GL_CRD_FLAGS                   ; 1199
                0000FE00  9F  00000000' EF  D0 0001C       MOVL     DS$L_SAVED_DSA_FLAGS, @#^X0000FE00    ; 1205
                          62  00000000' EF  D0 00027       MOVL     DS$L_SAVED_DS_FLAGS, DS$GL_FLAGS      ; 1206
                0000FE01  9F      0708  8F  AA 0002E       BICW2    #1800, @#^X0000FE01                   ; 1216
                          00000000G     EF  16 00037       JSB      DS_CLEANUP                            ; 1221
                          62            02  8A 0003D       BICB2    #2, DS$GL_FLAGS                       ; 1227
                          00000204      9F  D4 00040       CLRL     @#^X00000204                          ; 1228
                          00000000G     EF  16 00046       JSB      DSR$CHECK_MENUTEST_OFF_SET            ; 1232
                          09            50  E8 0004C       BLBS     R0, 1$                                ;
```

```
                                00000000G  EF  16 0004F         JSB     DSR$CHECK_AUTOTEST_OFF_SET                      ; 1233
                                        03  50  E9 00055         BLBC    R0, 2$                                          ;
                                            00 00058 1$:        HALT                                                    ; 1237
                                            FD  11 00059         BRB     1$                                              ;
                                00000000G   EF  16 0005B 2$:     JSB     BEGIN_BLISS                                     ; 1245
                                            04 00061             RET                                                     ; 1247

; Routine Size:  98 bytes,    Routine Base:  CODE + 03C3

;    1248  1     %SBTTL 'CRD_Error Routine'
;    1249  1
;    1250  1     ROUTINE CRD_Error (Error_Type, More_Info, Memory_Was_Allocated) : NOVALUE =
;    1251  1
;    1252  1     !++
;    1253  1     ! Functional Description:
;    1254  1     !
;    1255  1     !       Prints an error message, and optionally (depending on the Error_Type) unloads the CRD image.
;    1256  1     !
;    1257  1     ! Formal Parameters:
;    1258  1     !
;    1259  1     !       Error_Type:           The reason for the error.
;    1260  1     !       More_Info:            More information on the Error_Type (status values, etc.).
;    1261  1     !       Memory_Was_Allocated: Either K_Region_Expanded or K_Region_Not_Expanded.
;    1262  1     !
;    1263  1     ! Implicit Inputs:
;    1264  1     !
;    1265  1     !       DS$A_Auto_Or_Menu
;    1266  1     !
;    1267  1     ! Side Effects:
;    1268  1     !
;    1269  1     !       Prints an error message, and optionally (depending on the Error_Type) Unloads the CRD image.
;    1270  1     !
;    1271  1     ! Completion Codes:
;    1272  1     !
;    1273  1     !       None
;    1274  1     !--
;    1275  2     BEGIN                                   ! Routine CRD_Error
;    1276  2             REGISTER
;    1277  2                     CRD_Vectors     : REF VECTOR [, BYTE, SIGNED],
;    1278  2                     Save_DSA_Flags,
;    1279  2                     Return_Now,
;    1280  2                     Error_Message;
;    1281  2
;    1282  2             Module_Debug ('CRD_Error - Start:');                                                    ![07
;    1283  2
;    1284  2             CRD_Vectors = .DS$Q_CRD_Image_Desc [K_Image_Ptr];
;    1285  2
;    1286  2             Save_DSA_Flags = .DSA$GL_Flags; ! Save them locally
;    1287  2
;    1288  2             DSA$V_CRD_MenuTest_Off = Off;
;    1289  2                                     !                                                               [10]
;    1290  2             DSA$V_CRD_MenuTest_On = Off;
;    1291  2                                     !                                                               [10]
;    1292  2             DSA$V_CRD_AutoTest_Off = Off;
;    1293  2                                     !                                                               [10]
;    1294  2             DSA$V_Binary       = Off;
```

```
;    1295  2
;    1296  2              Error_Message = 0;              ! Initialize to point to zer0
;    1297  2              Return_Now    = False;
;    1298  2
;    1299  2              CASE .Error_Type FROM 0 TO K_Last_Error_Type - 1 OF
;    1300  2                      SET
;    1301  2                      [K_MM_Is_On]:
;    1302  2                              Error_Message = T_MM_On;
;    1303  2
;    1304  2                      [K_No_DSA_Bit_Set]:
;    1305  2                              IF (.DSA$V_CRD_Trace) THEN
;    1306  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_No_DSA_Bit_Set, .More_Info)
;    1307  2                              ELSE
;    1308  2                                      Error_Message = T_Unknown_Error;
;    1309  2
;    1310  2                      [K_Wrong_CPU]:
;    1311  2                              IF (.DSA$V_CRD_Trace) THEN
;    1312  3                                      BEGIN
;    1313  3                                      MAP
;    1314  3                                              More_Info : BYTE;
;    1315  3
;    1316  3                                      LOCAL
;    1317  3                                              CPU_Type;
;    1318  3
;    1319  3                                      IF (.More_Info EQL SID_780) THEN
;    1320  3                                              CPU_Type = T_CPU_780
;    1321  3                                      ELSE IF (.More_Info EQL SID_750) THEN
;    1322  3                                              CPU_Type = T_CPU_750
;    1323  3                                      ELSE
;    1324  3                                              CPU_Type = T_CPU_Unknown;
;    1325  3
;    1326  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Wrong_CPU, .CPU_Type);
;    1327  3                                      END
;    1328  2                              ELSE
;    1329  2                                      Error_Message = T_Invalid_CPU;
;    1330  2
;    1331  2                      [K_Determine_Size]:
;    1332  2                              IF (.DSA$V_CRD_Trace) THEN
;    1333  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Determine_Size, .More_Info)
;    1334  2                              ELSE
;    1335  2                                      IF (.More_Info EQL RMS$_FNF) THEN
;    1336  2                                              Error_Message = T_File_Not_Found
;    1337  2                                      ELSE IF (NOT .More_Info) THEN
;    1338  2                                              Error_Message = T_Unknown_Error;
;    1339  2
;    1340  2                      [K_Expand_Region]:
;    1341  2                              IF (.DSA$V_CRD_Trace) THEN
;    1342  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Expand_Region, .More_Info)
;    1343  2                              ELSE
;    1344  2                                      Error_Message = T_Unknown_Error;
;    1345  2
;    1346  2                      [K_Load_File]:
;    1347  2                              IF (.DSA$V_CRD_Trace) THEN
;    1348  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Load_File, .More_Info)
;    1349  2                              ELSE
;    1350  2                                      Error_Message = T_Unknown_Error;
;    1351  2
```

```
;    1352  2                    [K_CRD_Internal]:
;    1353  3                            BEGIN
;    1354  3                            !+
;    1355  3                            ! This check must be made to CRD_Trace, not to CRD_Debug, because the CRD_Debug          !E07
;    1356  3                            ! location may not contain the 0 or 1 that is used to indicate debugging mode.          !E07
;    1357  3                            ! It may contain a CRD dispatch vector (i.e., the vectors may still be switched).       !E07
;    1358  3                            !-
;    1359  3                            IF (.CRD$V_CRD_Debug) THEN                                                            !E07
;    1360  3                                    Internal_Error ()                                                             !E07
;    1361  3                            ELSE
;    1362  3                                    Error_Message = T_Corrupted_Error;
;    1363  3
;    1364  3                            Return_Now = True;
;    1365  3                            END;
;    1366  2
;    1367  2                    [K_Corrupted]:
;    1368  3                            IF (.DSA$V_CRD_Trace) THEN
;    1369  3                                    $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Corrupted)
;    1370  3                            ELSE
;    1371  3                                    Error_Message = T_Corrupted_Error;
;    1372  2
;    1373  2                    [K_Incompatible]:
;    1374  3                            IF (.DSA$V_CRD_Trace) THEN
;    1375  3                                    $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Incompatible)
;    1376  2                            ELSE
;    1377  2                                    Error_Message = T_Incompatibility_Error;
;    1378  2
;    1379  2                    [K_Not_Contracted]:
;    1380  3                            BEGIN
;    1381  3                            !+
;    1382  3                            ! If the contraction failed, then we must print an error message, call the
;    1383  3                            ! MapFree routine to map all the free areas of memory, and then exit CRD.
;    1384  3                            !-
;    1385  3
;    1386  3                            IF (.DSA$V_CRD_Trace) THEN
;    1387  3                                    $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Not_Contracted, .More_Info)
;    1388  3                            ELSE
;    1389  3                                    $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Contraction_Error);
;    1390  3
;    1391  3                            MapFree ();     ! Do this to Contract the region
;    1392  3                            CRD_Exit ();    ! Exit from here
;    1393  3                            END;
;    1394  2                    TES;
;    1395  2
;    1396  2            IF (.Error_Message NEQ 0) THEN
;    1397  2                    $Print (DS$K_Type_General_Error, DS$K_PrintF, .Error_Message, .DS$A_Auto_Or_Menu);
;    1398  2
;    1399  2            IF (.Return_Now) THEN
;    1400  3                    BEGIN
;    1401  3                    DSA$GL_Flags = .Save_DSA_Flags;
;    1402  3                    RETURN;                 ! All we had to do was to print an error message
;    1403  3                    END;
;    1404  2
;    1405  2            !+
;    1406  2            ! If it was Auto Test that was running (the RPB bit must be set if Auto Test was running),
;    1407  2            ! then print the final three lines of output.
;    1408  2            !-
```

```
;    1409  2
;    1410  2              IF (DSR$Check_AutoTest_Off_Set ()) THEN
;    1411  2                                              !                                                   [10]
;    1412  3                  BEGIN
;    1413  3                  $Print (DS$K_Type_General_Error, DS$K_PrintF, T_AUTO_Abort_CRD);
;    1414  3                  $Print (DS$K_Type_General_Error, DS$K_PrintF, T_AUTO_End_Message);
;    1415  3                  $Print (DS$K_Type_General_Error, DS$K_PrintF, T_AUTO_Exit_To_Console);
;    1416  3                  END;
;    1417  2
;    1418  2              IF (.Memory_Was_Allocated) THEN
;    1419  2                  DSR$Unload_CRD (0);     ! This should not return to here
;    1420  2
;    1421  2              DSA$GL_Flags = .Save_DSA_Flags; ! Restore the (locally saved) DSA flag settings
;    1422  2
;    1423  2              Module_Debug ('CRD_Error - End:');                                                  !E07
;    1424  2
;    1425  2              CRD_Exit ();                    ! Exit CRD
;    1426  2              RETURN;
;    1427  1      END;                                    ! Routine CRD_Error



                                                OFFC 00000 CRD_ERROR:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 1250
                                    57       99   AF  9E 00002          MOVAB   CRD_EXIT, R7                  ;
                                    56 00000000G  9F  9E 00006          MOVAB   @#DSX$PRINT, R6               ;
                                    55 00000000'  EF  9E 0000D          MOVAB   T_CORRUPTED_ERROR, R5         ;
                                    50 00000000'  EF  D0 00014          MOVL    DS$Q_CRD_IMAGE_DESC+4, CRD_VECTORS ; 1284
                                    52 0000FE00   9F  D0 0001B          MOVL    @#^X0000FE00, SAVE_DSA_FLAGS  ; 1286
                          0000FE00  9F 00050802   8F  CA 00022          BICL2   #329730, @#^X0000FE00         ; 1290
                                                53  7C 0002D          CLRQ    RETURN_NOW                    ; 1297
                               09               00  04 AC  CF 0002F          CASEL   ERROR_TYPE, #0, #9            ; 1299
                0089           0061          0028      001A      00034 1$:   .WORD   3$-1$,-
                00E4           00D1          0089      009A      0003C       4$-1$,-
                                             0014      00CF      00044       9$-1$,-
                                                                             13$-1$,-
                                                                             14$-1$,-
                                                                             17$-1$,-
                                                                             21$-1$,-
                                                                             23$-1$,-
                                                                             27$-1$,-
                                                                             2$-1$
                                    54       31   A5  9E 00048 2$:    MOVAB   T_MM_ON, ERROR_MESSAGE          ; 1302
                                                  67  11 0004C          BRB     1T$
                          78 0000FE02   9F          01  E1 0004E 3$:    BBC     #1, @#^X0000FE02, 14$         ; 1305
                                          08      AC  DD 00056          PUSHL   MORE_INFO                     ; 1306
                                        0150      C5  9F 00059          PUSHAB  T_NO_DSA_BIT_SET
                                                  7E  11 0005D          BRB     15$
                          27 0000FE02   9F          01  E1 0005F 4$:    BBC     #1, @#^X0000FE02, 8$          ; 1311
                                          01      08  AC  91 00067          CMPB    MORE_INFO, #1             ; 1319
                                                  07  12 0006B          BNEQ    5$
                                          50    01B6  C5  9E 0006D          MOVAB   T_CPU_780, CPU_TYPE           ; 1320
                                                  12  11 00072          BRB     7$
                                          02      08  AC  91 00074 5$:    CMPB    MORE_INFO, #2             ; 1321
                                                  07  12 00078          BNEQ    6$
```

```
                                50      01C3    C5  9E  0007A          MOVAB   T_CPU_750, CPU_TYPE                          ; 1322
                                                05  11  0007F          BRB     7$                                          ;
                                50      01D0    C5  9E  00081  6$:    MOVAB   T_CPU_UNKNOWN, CPU_TYPE                     ; 1324
                                                50  DD  00086  7$:    PUSHL   CPU_TYPE                                    ; 1326
                                        0180    C5  9F  00088          PUSHAB  T_WRONG_CPU                                 ;
                                                4F  11  0008C          BRB     15$                                         ;
                                54      FF0F    C5  9E  0008E  8$:    MOVAB   T_INVALID_CPU, ERROR_MESSAGE                ; 1329
                                                6E  11  00093          BRB     20$                                         ; 1311
                    09 0000FE02 9F              01  E1  00095  9$:    BBC     #1, @#^X0000FE02, 10$                       ; 1332
                                          08    AC  DD  0009D          PUSHL   MORE_INFO                                   ; 1333
                                        01DB    C5  9F  000A0          PUSHAB  T_DETERMINE_SIZE                            ;
                                                37  11  000A4          BRB     15$                                         ;
                       00018292 8F        08    AC  D1  000A6  10$:    CMPL    MORE_INFO, #98962                           ; 1335
                                                07  12  000AE          BNEQ    12$                                         ;
                                54      FF3E    C5  9E  000B0          MOVAB   T_FILE_NOT_FOUND, ERROR_MESSAGE             ; 1336
                                                7A  11  000B5  11$:    BRB     26$                                         ;
                                76        08    AC  E8  000B7  12$:    BLBS    MORE_INFO, 26$                              ; 1337
                                                29  11  000BB          BRB     16$                                         ; 1338
                    21 0000FE02 9F              01  E1  000BD  13$:    BBC     #1, @#^X0000FE02, 16$                       ; 1341
                                          08    AC  DD  000C5          PUSHL   MORE_INFO                                   ; 1342
                                        0222    C5  9F  000C8          PUSHAB  T_EXPAND_REGION                             ;
                                                0F  11  000CC          BRB     15$                                         ;
                    10 0000FE02 9F              01  E1  000CE  14$:    BBC     #1, @#^X0000FE02, 16$                       ; 1347
                                          08    AC  DD  000D6          PUSHL   MORE_INFO                                   ; 1348
                                        025B    C5  9F  000D9          PUSHAB  T_LOAD_FILE                                 ;
                                                01  DD  000DD  15$:    PUSHL   #1                                          ;
                                                03  DD  000DF          PUSHL   #3                                          ;
                                66              04  FB  000E1          CALLS   #4, DSX$PRINT                               ;
                                                79  11  000E4          BRB     30$                                         ;
                                54      FF71    C5  9E  000E6  16$:    MOVAB   T_UNKNOWN_ERROR, ERROR_MESSAGE              ; 1350
                                                72  11  000EB          BRB     30$                                         ; 1347
                                09 00000000G    EF  E9  000ED  17$:    BLBC    DS$GL_CRD_FLAGS, 18$                        ; 1359
                       00000000V EF             00  FB  000F4          CALLS   #0, INTERNAL_ERROR                          ; 1360
                                                03  11  000FB          BRB     19$                                         ;
                                54              65  9E  000FD  18$:    MOVAB   T_CORRUPTED_ERROR, ERROR_MESSAGE            ; 1362
                                53              01  D0  00100  19$:    MOVL    #1, RETURN_NOW                              ; 1364
                                                5A  11  00103  20$:    BRB     30$                                         ; 1299
                    06 0000FE02 9F              01  E1  00105  21$:    BBC     #1, @#^X0000FE02, 22$                       ; 1368
                                        0357    C5  9F  0010D          PUSHAB  T_CORRUPTED                                 ; 1369
                                                11  11  00111          BRB     24$                                         ;
                                54              65  9E  00113  22$:    MOVAB   T_CORRUPTED_ERROR, ERROR_MESSAGE            ; 1371
                                                47  11  00116          BRB     30$                                         ; 1368
                    0D 0000FE02 9F              01  E1  00118  23$:    BBC     #1, @#^X0000FE02, 25$                       ; 1374
                                        03A0    C5  9F  00120          PUSHAB  T_INCOMPATIBLE                              ; 1375
                                                01  DD  00124  24$:    PUSHL   #1                                          ;
                                                03  DD  00126          PUSHL   #3                                          ;
                                66              03  FB  00128          CALLS   #3, DSX$PRINT                               ;
                                                32  11  0012B          BRB     30$                                         ;
                                54        DF    A5  9E  0012D  25$:    MOVAB   T_INCOMPATIBILITY_ERROR, ERROR_MESSAGE      ; 1377
                                                2C  11  00131  26$:    BRB     30$                                         ; 1374
                    10 0000FE02 9F              01  E1  00133  27$:    BBC     #1, @#^X0000FE02, 28$                       ; 1386
                                          08    AC  DD  0013B          PUSHL   MORE_INFO                                   ; 1387
                                        0321    C5  9F  0013E          PUSHAB  T_NOT_CONTRACTED                            ;
                                                01  DD  00142          PUSHL   #1                                          ;
                                                03  DD  00144          PUSHL   #3                                          ;
                                66              04  FB  00146          CALLS   #4, DSX$PRINT                               ;
                                                0A  11  00149          BRB     29$                                         ;
```

```
                                                B4    A5  9F 0014B 28$:    PUSHAB  T_CONTRACTION_ERROR                     ; 1389
                                                      01  DD 0014E         PUSHL   #T                                      ;
                                                      03  DD 00150         PUSHL   #3                                      ;
                                             66       03  FB 00152         CALLS   #3, DSX$PRINT                           ;
                                 00000000G   EF       00  FB 00155 29$:    CALLS   #0, MAPFREE                             ; 1391
                                             67       00  FB 0015C         CALLS   #0, CRD_EXIT                            ; 1392
                                                      54  D5 0015F 30$:    TSTL    ERROR_MESSAGE                           ; 1396
                                                      0F  13 00161         BEQL    31$                                     ;
                                      00000000'  EF       DD 00163         PUSHL   DS$A_AUTO_OR_MENU                       ; 1397
                                                      54  DD 00169         PUSHL   ERROR_MESSAGE                           ;
                                                      01  DD 0016B         PUSHL   #1                                      ;
                                                      03  DD 0016D         PUSHL   #3                                      ;
                                             66       04  FB 0016F         CALLS   #4, DSX$PRINT                           ;
                                             08       53  E9 00172 31$:    BLBC    RETURN_NOW, 32$                         ; 1399
                                 0000FE00    9F       52  D0 00175         MOVL    SAVE_DSA_FLAGS, @#^X0000FE00            ; 1401
                                                      04  0017C           RET                                             ; 1400
                                      00000000G  EF   16  0017D 32$:    JSB     DSR$CHECK_AUTOTEST_OFF_SET              ; 1410
                                             21       50  E9 00183         BLBC    R0, 33$                                 ;
                                                00B8  C5  9F 00186         PUSHAB  T_AUTO_ABORT_CRD                        ; 1413
                                                      01  DD 0018A         PUSHL   #T                                      ;
                                                      03  DD 0018C         PUSHL   #3                                      ;
                                             66       03  FB 0018E         CALLS   #3, DSX$PRINT                           ;
                                                00E3  C5  9F 00191         PUSHAB  T_AUTO_END_MESSAGE                      ; 1414
                                                      01  DD 00195         PUSHL   #T                                      ;
                                                      03  DD 00197         PUSHL   #3                                      ;
                                             66       03  FB 00199         CALLS   #3, DSX$PRINT                           ;
                                                010F  C5  9F 0019C         PUSHAB  T_AUTO_EXIT_TO_CONSOLE                  ; 1415
                                                      01  DD 001A0         PUSHL   #T                                      ;
                                                      03  DD 001A2         PUSHL   #3                                      ;
                                             66       03  FB 001A4         CALLS   #3, DSX$PRINT                           ;
                                             07    0C  AC  E9 001A7 33$:    BLBC    MEMORY_WAS_ALLOCATED, 34$               ; 1418
                                                   7E  D4 001AB           CLRL    -(SP)                                   ; 1419
                                        FE68    C7       01  FB 001AD         CALLS   #1, DSR$UNLOAD_CRD                      ;
                                    0000FE00    9F       52  D0 001B2 34$:    MOVL    SAVE_DSA_FLAGS, @#^X0000FE00            ; 1421
                                             67       00  FB 001B9         CALLS   #0, CRD_EXIT                            ; 1425
                                                      04  001BC           RET                                             ; 1427

; Routine Size:  445 bytes,    Routine Base:  CODE + 0425


;    1428  1     %SBTTL  'Internal_Error Routine'
;    1429  1
;    1430  1     ROUTINE Internal_Error : NOVALUE =                                                                  !C07
;    1431  1     !++                                                                                                 ! V
;    1432  1     ! Functional Description:
;    1433  1     !
;    1434  1     !       An internal error has occurred. The operator has already been notified. If the Trace or the Debug bits
;    1435  1     !       are set, print this 'error dump' that shows the state of CRD at the time of the internal error. This will
;    1436  1     !       help the Diagnostic Tools group people tremendously in debugging CRD. Hopefully.
;    1437  1     !
;    1438  1     ! Formal Parameters:
;    1439  1     !
;    1440  1     !       None
;    1441  1     !
;    1442  1     ! Implicit Inputs:
;    1443  1     !
;    1444  1     !       DS$Q_CRD_Image_Desc: This points to the start of the CRD image (in memory). The dispatch vectors (or, in
```

```
;    1445  1      !                                this case, the debug vectors) are located at the start of the image.
;    1446  1      !
;    1447  1      ! Side Effects:
;    1448  1      !
;    1449  1      !       Prints some debugging information.
;    1450  1      !
;    1451  1      ! Completion Codes:
;    1452  1      !
;    1453  1      !       None
;    1454  1      !--
;    1455  2      BEGIN
;    1456  2              MAP
;    1457  2                      DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE];
;    1458  2
;    1459  2              REGISTER
;    1460  2                      Stat;
;    1461  2
;    1462  2              BIND
;    1463  2                      Debug_Vectors = (.DS$Q_CRD_Image_Desc [K_Image_Ptr] + 4) : VECTOR [, LONG];
;    1464  2
;    1465  2              BIND
;    1466  2                      Error_Code              = Debug_Vectors [1],
;    1467  2                      TypeCode                = Debug_Vectors [2],
;    1468  2                      Length                  = Debug_Vectors [3],
;    1469  2                      Address                 = Debug_Vectors [4],
;    1470  2                      Current_State_Table     = Debug_Vectors [5],
;    1471  2                      MM_Current_State        = Debug_Vectors [6],
;    1472  2                      TQ_Current_State        = Debug_Vectors [7],
;    1473  2                      HS_Current_State        = Debug_Vectors [8],
;    1474  2                      Previous_State          = Debug_Vectors [9],
;    1475  2                      DSA_Flags               = Debug_Vectors [10],
;    1476  2                      DS_Flags                = Debug_Vectors [11],
;    1477  2                      Current_DDB_Ptr         = Debug_Vectors [12],
;    1478  2                      Current_HSB_Ptr         = Debug_Vectors [13],
;    1479  2                      Current_HSB_Numb        = Debug_Vectors [14],
;    1480  2                      Current_TQ_Entry        = Debug_Vectors [15],
;    1481  2                      Start_Of_CRD_Image      = Debug_Vectors [16],
;    1482  2                      CRD_Version_Number      = ((.Debug_Vectors [16]) <8, 8, 0>),
;    1483  2                      VDS_Version_Number      = DS$AL_DS_Dispatch_Vectors [1];
;    1484  2
;    1485  2              MACRO
M   1486  2                      Error_Type (Error_Type_Name) =
M   1487  2                              [- %NAME ?'CRD$K_', Error_Type_Name)]:
M   1488  2                                      $Print (DS$K_Type_General_Error, DS$K_PrintF,
M   1489  2                                              $ASCIC ('Error Type = ', Error_Type_Name, '!/'))
;    1490  2                      %;
;    1491  2
;    1492  2              Module_Debug ('Internal_Error - Start:');
;    1493  2
P   1494  2              $Print (DS$K_Type_General_Error, DS$K_PrintF,
;    1495  2                      $ASCIC ('!/!/************************ CRD Internal Error ************************!/!/'));
;    1496  2
;    1497  2              ! Print the type of error
;    1498  2
;    1499  2              CASE -.Error_Code FROM 1 TO (-CRD$K_Last_Internal_Error - 1) OF
;    1500  2                      SET
;    1501  2                      Error_Type ('Allocation_Error');
```

```
;    1502  2                    Error_Type ('Action_Range_Error');
;    1503  2                    Error_Type ('Action_EQL_Do_Nothing');
;    1504  2                    Error_Type ('Bad_Typecode_Error');
;    1505  2                    Error_Type ('Data_Length_Error');
;    1506  2                    Error_Type ('MM_Internal_Error');
;    1507  2                    Error_Type ('TQ_Internal_Error');
;    1508  2                    Error_Type ('HS_Internal_Error');
;    1509  2                    Error_Type ('Bad_Action_Number');
;    1510  2                    Error_Type ('Load_Sup_File');
;    1511  2                    Error_Type ('Create_HST');
;    1512  2                    [INRANGE]:
;  P 1513  2                            $Print (DS$K_Type_General_Error, DS$K_Printf,
;    1514  2                                    $ASCIC ('Error Type = unknown (!SL)!/'), .Error_Code);
;    1515  2
;    1516  2                    [OUTRANGE]:
;  P 1517  2                            $Print (DS$K_Type_General_Error, DS$K_Printf,
;    1518  2                                    $ASCIC ('Error Type = out of range (!SL)!/'), .Error_Code);
;    1519  2                    TES;
;    1520  2
;    1521  2            ! Print the typecode name
;    1522  2
;    1523  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('Typecode !2* = '));
;    1524  2            DSR$Print_TypeCode_Name (.TypeCode);
;    1525  2
;    1526  2            ! Print the Length of the message being printed at the time the internal occurred. Note that this will
;    1527  2            ! not always be a length, as in the internal errors in the MENTSTINI module.
;    1528  2
;    1529  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('Length !4* = !UL!/'), .Length);
;    1530  2
;    1531  2            ! Print the address - see previous comment.
;    1532  2
;    1533  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('Address !3* = !XL(X)!/'), .Address);
;    1534  2
;    1535  2            ! Print the DSA and the DS flags.
;    1536  2
;    1537  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('DSA flags !1* = !XL(X)!/'), .DSA_Flags);
;    1538  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('DS flags !2* = !XL(X)!/'), .DS_Flags);
;    1539  2
;    1540  2            ! Print the state table information
;    1541  2
;    1542  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('!/State Table !10* State Number'));
;    1543  2            $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('!/------------ !10* ------------!/'));
;    1544  2
;    1545  2            State = .Current_State_Table;   ! Start at the current state table
;    1546  2            DO
;    1547  3                    BEGIN
;    1548  3                    CASE .State FROM MEN$K_MM_State_Table TO MEN$K_HS_State_Table OF
;    1549  3                            SET
;    1550  3                            [MEN$K_MM_State_Table]:
;    1551  4                                    BEGIN
;  P 1552  4                                    $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('Main Menu !17* !2UL!/'),
;    1553  4                                            .MM_Current_State);
;    1554  3                                    END;
;    1555  3
;    1556  3                            [MEN$K_TQ_State_Table]:
;    1557  4                                    BEGIN
;  P 1558  4                                    $Print (DS$K_Type_General_Error, DS$K_Printf, $ASCIC ('Test Queue !16* !2UL!/'),
```

```
;    1559  4                                          .TQ_Current_State);
;    1560  3                                  END;
;    1561  3
;    1562  3                              [MEN$K_HS_State_Table]:
;    1563  4                                  BEGIN
; P  1564  4                                  $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Hardware Support !10* !2UL!/'
;    1565  4                                          .HS_Current_State);
;    1566  3                                  END;
;    1567  3                              TES;
;    1568  3
;    1569  3                              State = (.State + 1) MOD 3;
;    1570  3                      END                 ! DO . . . UNTIL
;    1571  2              UNTIL (.State EQL .Current_State_Table);
;    1572  2
;    1573  2              ! Print the previous state number
;    1574  2
;    1575  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Previous state !12* !2UL!/!/'), .Previous_State);
;    1576  2
;    1577  2              ! Print the DDB and HSB (Device Data Block and Hardware Support Block) pointers
;    1578  2
;    1579  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current TQ entry    = !UL!/'),    .Current_TQ_Entry);
;    1580  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current DDB address = !XL(X)!/'), .Current_DDB_Ptr);
;    1581  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current HSB number  = !UL!/'),    .Current_HSB_Numb);
;    1582  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current HSB address = !XL(X)!/!/'), .Current_HSB_Ptr)
;    1583  2
;    1584  2              ! Now, print the remaining debug vectors. The remaining ones are either 0 or contain a PC. Print only
;    1585  2              ! those that are PC's.
;    1586  2
; P  1587  2              $Print (DS$K_Type_General_Error, DS$K_PrintF,
;    1588  2                      $ASCIC ('PCs from the call frame stack (starting with most recent):!/'));
;    1589  2
;    1590  2              INCR Vector FROM 17 to (CRD$K_Total_Numb_Vectors - 1) DO
;    1591  2                      IF (.Debug_Vectors [.Vector] NEQ 0) THEN
; P  1592  2                              $Print (DS$K_Type_General_Error, DS$K_PrintF,
;    1593  3                                      $ASCIC ('  PC from call frame [!UL] = !XL!/'), (.Vector-17), .Debug_Vectors [.Vector
;    1594  2                      ELSE
;    1595  2                              EXITLOOP;
;    1596  2
;    1597  2              Module_Debug ('Internal_Error - End:');                                                          ! ^
;    1598  1      END;                                    ! End of Internal_Error                                         !C07


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2F 21 2F 21 4C 006EB P.ABN:   .ASCII  \L!/!/********************** CRD Intern\
20 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 006FA
                  6E 72 65 74 6E 49 20 44 52 43 00709
2A 2A 2A 2A 2A 2A 20 72 6F 72 72 45 20 6C 61 00713          .ASCII  \al Error **********************!/!/\
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 00722
                     2F 21 2F 21 2A 2A 2A 00731
41 20 3D 20 65 70 79 54 20 72 6F 72 72 45 1F 00738 P.ABO:   .ASCII  <31>\Error Type = Allocation_Error!/\
72 6F 72 72 45 5F 6E 6F 69 74 61 63 6F 6C 6C 00747
                                       2F 21 00756
41 20 3D 20 65 70 79 54 20 72 6F 72 72 45 21 00758 P.ABP:   .ASCII  \!Error Type = Action_Range_Erro!/\
72 72 45 5F 65 67 6E 61 52 5F 6E 6F 69 74 63 00767
                                 2F 21 72 6F 00776
```

```
41 20 3D 20 65 70 79 54 20 72 6F 72 72 45 24 0077A P.ABQ:  .ASCII  \$Error Type = Action_EQL_Do_Nothing!/\
6F 4E 5F 6F 44 5F 4C 51 45 5F 6E 6F 69 74 63 00789
                  2F 21 67 6E 69 68 74 00798
42 20 3D 20 65 70 79 54 20 72 6F 72 72 45 21 0079F P.ABR:  .ASCII  \!Error Type = Bad_Typecode_Error!/\
72 72 45 5F 65 64 6F 63 65 70 79 54 5F 64 61 007AE
                        2F 21 72 6F 007BD
44 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 007C1 P.ABS:  .ASCII  \ Error Type = Data_Length_Error!/\
6F 72 72 45 5F 68 74 67 6E 65 4C 5F 61 74 61 007D0
                              2F 21 72 007DF
4D 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 007E2 P.ABT:  .ASCII  \ Error Type = MM_Internal_Error!/\
6F 72 72 45 5F 6C 61 6E 72 65 74 6E 49 5F 4D 007F1
                              2F 21 72 00800
54 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 00803 P.ABU:  .ASCII  \ Error Type = TQ_Internal_Error!/\
6F 72 72 45 5F 6C 61 6E 72 65 74 6E 49 5F 51 00812
                              2F 21 72 00821
48 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 00824 P.ABV:  .ASCII  \ Error Type = HS_Internal_Error!/\
6F 72 72 45 5F 6C 61 6E 72 65 74 6E 49 5F 53 00833
                              2F 21 72 00842
42 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 00845 P.ABW:  .ASCII  \ Error Type = Bad_Action_Number!/\
65 62 6D 75 4E 5F 6E 6F 69 74 63 41 5F 64 61 00854
                              2F 21 72 00863
4C 20 3D 20 65 70 79 54 20 72 6F 72 72 45 1C 00866 P.ABX:  .ASCII  <28>\Error Type = Load_Sup_File!/\
   2F 21 65 6C 69 46 5F 70 75 53 5F 64 61 6F 00875
43 20 3D 20 65 70 79 54 20 72 6F 72 72 45 19 00883 P.ABY:  .ASCII  <25>\Error Type = Create_HST!/\
                  2F 21 54 53 48 5F 65 74 61 65 72 00892
75 20 3D 20 65 70 79 54 20 72 6F 72 72 45 1C 0089D P.ABZ:  .ASCII  <28>\Error Type = unknown (!SL)!/\
   2F 21 29 4C 53 21 28 20 6E 77 6F 6E 6B 6E 008AC
6F 20 3D 20 65 70 79 54 20 72 6F 72 72 45 21 008BA P.ACA:  .ASCII  \!Error Type = out of range (!SL)!/\
53 21 28 20 65 67 6E 61 72 20 66 6F 20 74 75 008C9
                        2F 21 29 4C 008D8
3D 20 2A 32 21 20 65 64 6F 63 65 70 79 54 0F 008DC P.ACB:  .ASCII  <15>\Typecode !2* = \
                                          20 008EB
21 20 3D 20 2A 34 21 20 68 74 67 6E 65 4C 12 008EC P.ACC:  .ASCII  <18>\Length !4* = !UL!/\
                        2F 21 4C 55 008FB
20 3D 20 2A 33 21 20 73 73 65 72 64 64 41 16 008FF P.ACD:  .ASCII  <22>\Address !3* = !XL(X)!/\
                  2F 21 29 58 28 4C 58 21 0090E
20 2A 31 21 20 73 67 61 6C 66 20 41 53 44 18 00916 P.ACE:  .ASCII  <24>\DSA flags !1* = !XL(X)!/\
                  2F 21 29 58 28 4C 58 21 20 3D 00925
3D 20 2A 32 21 20 73 67 61 6C 66 20 53 44 17 0092F P.ACF:  .ASCII  <23>\DS flags !2* = !XL(X)!/\
                        2F 21 29 58 28 4C 58 21 20 0093E
20 65 6C 62 61 54 20 65 74 61 74 53 2F 21 1F 00947 P.ACG:  .ASCII  <31>\!/State Table !10* State Number\
62 6D 75 4E 20 65 74 61 74 53 20 2A 30 31 21 00956
                                    72 65 00965
20 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2F 21 21 00967 P.ACH:  .ASCII  \!!/---------- !10* ----------!/\
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 20 2A 30 31 21 00976
                              2F 21 2D 2D 00985
2A 37 31 21 20 75 6E 65 4D 20 6E 69 61 4D 15 00989 P.ACI:  .ASCII  <21>\Main Menu !17* !2UL!/\
                  2F 21 4C 55 32 21 20 00998
36 31 21 20 65 75 65 75 51 20 74 73 65 54 16 0099F P.ACJ:  .ASCII  <22>\Test Queue !16* !2UL!/\
                  2F 21 4C 55 32 21 20 2A 009AE
6F 70 70 75 53 20 65 72 61 77 64 72 61 48 1C 009B6 P.ACK:  .ASCII  <28>\Hardware Support !10* !2UL!/\
   2F 21 4C 55 32 21 20 2A 30 31 21 20 74 72 009C5
65 74 61 74 73 20 73 75 6F 69 76 65 72 50 1C 009D3 P.ACL:  .ASCII  <28>\Previous state !12* !2UL!/!/\
   2F 21 2F 21 4C 55 32 21 20 2A 32 31 21 20 009E2
74 6E 65 20 51 54 20 74 6E 65 72 72 75 43 18 009F0 P.ACM:  .ASCII  <27>\Current TQ entry    = !UL!/\
      2F 21 4C 55 21 20 3D 20 20 20 20 79 72 009FF
64 61 20 42 44 44 20 74 6E 65 72 72 75 43 1E 00A0C P.ACN:  .ASCII  <30>\Current DDB address = !XL(X)!/\
```

```
21 29 58 28 4C 58 21 20 3D 20 73 73 65 72 64 00A1B
                                  2F 00A2A
75 6E 20 42 53 48 20 74 6E 65 72 72 75 43 1B 00A2B P.ACO:  .ASCII  <27>\Current HSB number  = !UL!/\
   2F 21 4C 55 21 20 3D 20 20 72 65 62 6D 00A3A
64 61 20 42 53 48 20 74 6E 65 72 72 75 43 20 00A47 P.ACP:  .ASCII  \ Current HSB address = !XL(X)!/!/\
21 29 58 28 4C 58 21 20 3D 20 73 73 65 72 64 00A56
                                  2F 21 2F 00A65
63 20 65 68 74 20 6D 6F 72 66 20 73 43 50 3C 00A68 P.ACQ:  .ASCII  \<PCs from the call frame stack (starting\
6B 63 61 74 73 20 65 6D 61 72 66 20 6C 6C 61 00A77
                  67 6E 69 74 72 61 74 73 28 20 00A86
65 63 65 72 20 74 73 6F 6D 20 68 74 69 77 20 00A90          .ASCII  \ with most recent):!/\
                              2F 21 3A 29 74 6E 00A9F
6C 6C 61 63 20 6D 6F 72 66 20 43 50 20 20 22 00AA5 P.ACR:  .ASCII  \"  PC from call frame [!UL] = !XL!/\
20 3D 20 5D 4C 55 21 5B 20 65 6D 61 72 66 20 00AB4
                              2F 21 4C 58 21 00AC3

                                                          .PSECT  CODE,NOWRT,  SHR,2

                                                  003C 00000 INTERNAL_ERROR:
                                                             .WORD   Save R2,R3,R4,R5                    ; 1430
                                55 00000000G   9F   9E 00002 MOVAB   a#DSX$PRINT, R5
                                54 00000000'   EF   9E 00009 MOVAB   P.ABN, R4
                     53 00000000'   EF         04   C1 00010 ADDL3   #4, DS$Q_CRD_IMAGE_DESC+4, R3      ; 1463
                                               54   DD 00018 PUSHL   R4                                 ; 1495
                                               01   DD 0001A PUSHL   #1
                                               03   DD 0001C PUSHL   #3
                                65             03   FB 0001E CALLS   #3, DSX$PRINT
                                50         04  A3   CE 00021 MNEGL   4(R3), R0                        ; 1499
                             0A 01             50   CF 00025 CASEL   R0, #1, #10
0036                    0030    0028          0026      00029 1$:     .WORD   2$-1$,-
004E                    0048    0042          003C      00031                 3$-1$,-
                        0060    005A          0054      00039                 4$-1$,-
                                                                              5$-1$,-
                                                                              6$-1$,-
                                                                              7$-1$,-
                                                                              8$-1$,-
                                                                              9$-1$,-
                                                                              10$-1$,-
                                                                              11$-1$,-
                                                                              12$-1$
                                           04  A3   DD 0003F       PUSHL   4(R3)                       ; 1518
                                        01CF   C4   9F 00042       PUSHAB  P.ACA
                                               01   DD 00046       PUSHL   #1
                                               03   DD 00048       PUSHL   #3
                                65             04   FB 0004A       CALLS   #4, DSX$PRINT
                                               45   11 0004D       BRB     14$
                                           4D  A4   9F 0004F 2$:     PUSHAB  P.ABO                       ; 1501
                                               39   11 00052       BRB     13$
                                           6D  A4   9F 00054 3$:     PUSHAB  P.ABP                       ; 1502
                                               34   11 00057       BRB     13$
                                        008F   C4   9F 00059 4$:     PUSHAB  P.ABQ                       ; 1503
                                               2E   11 0005D       BRB     13$
                                        00B4   C4   9F 0005F 5$:     PUSHAB  P.ABR                       ; 1504
                                               28   11 00063       BRB     13$
                                        00D6   C4   9F 00065 6$:     PUSHAB  P.ABS                       ; 1505
```

```
                                  22  11 00069      BRB     13$                            ;
                          00F7    C4  9F 0006B 7$:  PUSHAB  P.ABT                          ; 1506
                                  1C  11 0006F      BRB     13$                            ;
                          0118    C4  9F 00071 8$:  PUSHAB  P.ABU                          ; 1507
                                  16  11 00075      BRB     13$                            ;
                          0139    C4  9F 00077 9$:  PUSHAB  P.ABV                          ; 1508
                                  10  11 0007B      BRB     13$                            ;
                          015A    C4  9F 0007D 10$: PUSHAB  P.ABW                          ; 1509
                                  0A  11 00081      BRB     13$                            ;
                          017B    C4  9F 00083 11$: PUSHAB  P.ABX                          ; 1510
                                  04  11 00087      BRB     13$                            ;
                          0198    C4  9F 00089 12$: PUSHAB  P.ABY                          ; 1511
                                  01  DD 0008D 13$: PUSHL   #1                             ;
                                  03  DD 0008F      PUSHL   #3                             ;
                  65              03  FB 00091      CALLS   #3, DSX$PRINT                  ;
                          01F1    C4  9F 00094 14$: PUSHAB  P.ACB                          ; 1523
                                  01  DD 00098      PUSHL   #1                             ;
                                  03  DD 0009A      PUSHL   #3                             ;
                  65              03  FB 0009C      CALLS   #3, DSX$PRINT                  ;
                            08    A3  DD 0009F      PUSHL   8(R3)                          ; 1524
     00000000V    EF              01  FB 000A2      CALLS   #1, DSR$PRINT_TYPECODE_NAME    ;
                            0C    A3  DD 000A9      PUSHL   12(R3)                         ; 1529
                          0201    C4  9F 000AC      PUSHAB  P.ACC                          ;
                                  01  DD 000B0      PUSHL   #1                             ;
                                  03  DD 000B2      PUSHL   #3                             ;
                  65              04  FB 000B4      CALLS   #4, DSX$PRINT                  ;
                            10    A3  DD 000B7      PUSHL   16(R3)                         ; 1533
                          0214    C4  9F 000BA      PUSHAB  P.ACD                          ;
                                  01  DD 000BE      PUSHL   #1                             ;
                                  03  DD 000C0      PUSHL   #3                             ;
                  65              04  FB 000C2      CALLS   #4, DSX$PRINT                  ;
                            28    A3  DD 000C5      PUSHL   40(R3)                         ; 1537
                          022B    C4  9F 000C8      PUSHAB  P.ACE                          ;
                                  01  DD 000CC      PUSHL   #1                             ;
                                  03  DD 000CE      PUSHL   #3                             ;
                  65              04  FB 000D0      CALLS   #4, DSX$PRINT                  ;
                            2C    A3  DD 000D3      PUSHL   44(R3)                         ; 1538
                          0244    C4  9F 000D6      PUSHAB  P.ACF                          ;
                                  01  DD 000DA      PUSHL   #1                             ;
                                  03  DD 000DC      PUSHL   #3                             ;
                  65              04  FB 000DE      CALLS   #4, DSX$PRINT                  ;
                          025C    C4  9F 000E1      PUSHAB  P.ACG                          ; 1542
                                  01  DD 000E5      PUSHL   #1                             ;
                                  03  DD 000E7      PUSHL   #3                             ;
                  65              03  FB 000E9      CALLS   #3, DSX$PRINT                  ;
                          027C    C4  9F 000EC      PUSHAB  P.ACH                          ; 1543
                                  01  DD 000F0      PUSHL   #1                             ;
                                  03  DD 000F2      PUSHL   #3                             ;
                  65              03  FB 000F4      CALLS   #3, DSX$PRINT                  ;
                  52        14    A3  D0 000F7      MOVL    20(R3), STATE                  ; 1545
       02         00              52  CF 000FB 15$: CASEL   STATE, #0, #2                  ; 1548
     0018       000F        0006         000FF 16$: .WORD   17$-16$,-                      ;
                                                            18$-16$,-                      ;
                                                            19$-16$                        ;
                            18    A3  DD 00105 17$: PUSHL   24(R3)                         ; 1553
                          029E    C4  9F 00108      PUSHAB  P.ACI                          ;
                                  10  11 0010C      BRB     20$                            ;
```

```
                                                    1C   A3   DD 0010E 18$:    PUSHL   28(R3)                          ; 1559
                                                  02B4   C4   9F 00111          PUSHAB  P.ACJ                           ;
                                                         07   11 00115          BRB     20$                             ;
                                                    20   A3   DD 00117 19$:    PUSHL   32(R3)                          ; 1565
                                                  02C8   C4   9F 0011A          PUSHAB  P.ACK                           ;
                                                         01   DD 0011E 20$:    PUSHL   #1                              ;
                                                         03   DD 00120          PUSHL   #3                              ;
                                              65         04   FB 00122          CALLS   #4, DSX$PRINT                   ;
                                    7E    01    52         01   7A 00125          EMUL    #1, STATE, #1, -(SP)            ; 1569
                                    52    52    8E         03   7B 0012A          EDIV    #3, (SP)+, STATE, STATE         ;
                                          14    A3         52   D1 0012F          CMPL    STATE, 20(R3)                   ; 1571
                                                         C6   12 00133          BNEQ    15$                             ;
                                                    24   A3   DD 00135          PUSHL   36(R3)                          ; 1575
                                                  02E8   C4   9F 00138          PUSHAB  P.ACL                           ;
                                                         01   DD 0013C          PUSHL   #1                              ;
                                                         03   DD 0013E          PUSHL   #3                              ;
                                              65         04   FB 00140          CALLS   #4, DSX$PRINT                   ;
                                                    3C   A3   DD 00143          PUSHL   60(R3)                          ; 1579
                                                  0305   C4   9F 00146          PUSHAB  P.ACM                           ;
                                                         01   DD 0014A          PUSHL   #1                              ;
                                                         03   DD 0014C          PUSHL   #3                              ;
                                              65         04   FB 0014E          CALLS   #4, DSX$PRINT                   ;
                                                    30   A3   DD 00151          PUSHL   48(R3)                          ; 1580
                                                  0321   C4   9F 00154          PUSHAB  P.ACN                           ;
                                                         01   DD 00158          PUSHL   #1                              ;
                                                         03   DD 0015A          PUSHL   #3                              ;
                                              65         04   FB 0015C          CALLS   #4, DSX$PRINT                   ;
                                                    38   A3   DD 0015F          PUSHL   56(R3)                          ; 1581
                                                  0340   C4   9F 00162          PUSHAB  P.ACO                           ;
                                                         01   DD 00166          PUSHL   #1                              ;
                                                         03   DD 00168          PUSHL   #3                              ;
                                              65         04   FB 0016A          CALLS   #4, DSX$PRINT                   ;
                                                    34   A3   DD 0016D          PUSHL   52(R3)                          ; 1582
                                                  035C   C4   9F 00170          PUSHAB  P.ACP                           ;
                                                         01   DD 00174          PUSHL   #1                              ;
                                                         03   DD 00176          PUSHL   #3                              ;
                                              65         04   FB 00178          CALLS   #4, DSX$PRINT                   ;
                                                  037D   C4   9F 0017B          PUSHAB  P.ACQ                           ; 1588
                                                         01   DD 0017F          PUSHL   #1                              ;
                                                         03   DD 00181          PUSHL   #3                              ;
                                              65         03   FB 00183          CALLS   #3, DSX$PRINT                   ;
                                              52         11   DO 00186          MOVL    #17, VECTOR                     ; 1590
                                                       6342   D5 00189 21$:     TSTL    (R3)[VECTOR]                    ; 1591
                                                         15   13 0018C          BEQL    22$                             ;
                                                       6342   DD 0018E          PUSHL   (R3)[VECTOR]                    ; 1593
                                                         EF   A2 9F 00191       PUSHAB  -17(VECTOR)                     ;
                                                  038A   C4   9F 00194          PUSHAB  P.ACR                           ;
                                                         01   DD 00198          PUSHL   #1                              ;
                                                         03   DD 0019A          PUSHL   #3                              ;
                                              65         05   FB 0019C          CALLS   #5, DSX$PRINT                   ;
                                        E6    52         18   F3 0019F          AOBLEQ  #24, VECTOR, 21$                ; 1591
                                                         04   001A3 22$:        RET                                     ; 1598

; Routine Size:  420 bytes,    Routine Base:  CODE + 05E2

;   1599  1     %SBTTL 'DSR$Print_TypeCode_Name Routine'
```

```
;    1600  1
;    1601  1     GLOBAL ROUTINE DSR$Print_TypeCode_Name (TypeCode) : NOVALUE =                                    !E07
;    1602  1                                                                                                     ! V
;    1603  1     !++
;    1604  1     ! Functional Description:
;    1605  1     !
;    1606  1     !       Print a short message saying what a given typecode is used for.
;    1607  1     !
;    1608  1     ! Formal Input Parameters:
;    1609  1     !
;    1610  1     !       TypeCode: The typecode constant.
;    1611  1     !
;    1612  1     ! Formal Output Parameters:
;    1613  1     !
;    1614  1     !       None
;    1615  1     !
;    1616  1     ! Side Effects:
;    1617  1     !
;    1618  1     !       None
;    1619  1     !
;    1620  1     ! Completion Codes:
;    1621  1     !
;    1622  1     !       None
;    1623  1     !--
;    1624  2     BEGIN                                   ! Routine DSR$Print_TypeCode_Name
;    1625  2
;    1626  2             MACRO
;  M 1627  2                     TypeCode_Case (TypeCode_Name) =
;  M 1628  2                             [%NAME ('DS$K_Type_', TypeCode_Name)]:
;  M 1629  2                                     $Print (DS$K_Type_General, DS$K_PrintF, $ASCIC (TypeCode_Name, '!/'))
;    1630  2                     %;
;    1631  2
;  P 1632  2             CRD_Assert ((CRD$K_Numb_Typecodes EQL 38),
;  L 1633  2                     'You must make changes to the CASE statement below if you add more typecodes');
;  %PRINT:       ASSERT: The assertion is true.
;    1634  2
;    1635  2             CASE .TypeCode FROM 0 TO (CRD$K_Numb_TypeCodes - 1) OF
;    1636  2                     SET
;    1637  2                     TypeCode_Case ('General');
;    1638  2                     TypeCode_Case ('DS_Prompt');
;    1639  2                     TypeCode_Case ('User_Prompt');
;    1640  2                     TypeCode_Case ('General_Error');
;    1641  2                     TypeCode_Case ('Errsup');
;    1642  2                     TypeCode_Case ('Errsys');
;    1643  2                     TypeCode_Case ('Errhard');
;    1644  2                     TypeCode_Case ('Errsoft');
;    1645  2                     TypeCode_Case ('Errdev');
;    1646  2                     TypeCode_Case ('Error_Body');
;    1647  2                     TypeCode_Case ('Error_End');
;    1648  2                     TypeCode_Case ('Exception_Head');
;    1649  2                     TypeCode_Case ('Exception');
;    1650  2                     TypeCode_Case ('Err_Halt');
;    1651  2                     TypeCode_Case ('Summary');
;    1652  2                     TypeCode_Case ('Program_Start');
;    1653  2                     TypeCode_Case ('Program_End');
;    1654  2                     TypeCode_Case ('First_Pass');
;    1655  2                     TypeCode_Case ('No_Tests');
```

```
;    1656  2                      TypeCode_Case ('Abort_Test');
;    1657  2                      TypeCode_Case ('Abort_Program');
;    1658  2                      TypeCode_Case ('Command_Err');
;    1659  2                      TypeCode_Case ('Command_Out');
;    1660  2                      TypeCode_Case ('Program_Info');
;    1661  2                      TypeCode_Case ('Start_Err');
;    1662  2                      TypeCode_Case ('Sequence_Error');
;    1663  2                      TypeCode_Case ('CRD_AutoTest');
;    1664  2                      TypeCode_Case ('Errprep');
;    1665  2                      TypeCode_Case ('Param_Error');
;    1666  2                      TypeCode_Case ('DS_Start');
;    1667  2                      TypeCode_Case ('Script_Pnf');
;    1668  2                      TypeCode_Case ('Script_Skip');
;    1669  2                      TypeCode_Case ('Script_Prompt');
;    1670  2                      TypeCode_Case ('Script_Echo');
;    1671  2                      TypeCode_Case ('Qio_Nodriver');
;    1672  2                      TypeCode_Case ('Qio_Wrongver');
;    1673  2                      TypeCode_Case ('Qio_Invadp');
;    1674  2                      TypeCode_Case ('Start_List');
;    1675  2
;    1676  2                      [INRANGE]:
;    1677  2                              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Unknown typecode = !UL!/'), .Typecode
;    1678  2
;    1679  2                      [OUTRANGE]:
;    1680  2                              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Undefined typecode = !UL!/'), .TypeCo
;    1681  2                      TES;                                                                                            !
;    1682  1      END;                                    ! Routine DSR$Print_TypeCode_Name                                  !E07


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                                00AC8 P.ACS:    .ASCII  <9>\General!/\                                          ;
                                                00AD2 P.ACT:    .ASCII  <11>\DS_Prompt!/\                                       ;
                                                00ADE P.ACU:    .ASCII  <13>\User_Prompt!/\                                     ;
                                                00AEC P.ACV:    .ASCII  <15>\General_Error!/\                                   ;
                                                00AFB
                                                00AFC P.ACW:    .ASCII  <8>\Errsup!/\                                          ;
                                                00B05 P.ACX:    .ASCII  <8>\Errsys!/\                                          ;
                                                00B0E P.ACY:    .ASCII  <9>\Errhard!/\                                         ;
                                                00B18 P.ACZ:    .ASCII  <9>\Errsoft!/\                                         ;
                                                00B22 P.ADA:    .ASCII  <8>\Errdev!/\                                          ;
                                                00B2B P.ADB:    .ASCII  <12>\Error_Body!/\                                     ;
                                                00B38 P.ADC:    .ASCII  <11>\Error_End!/\                                      ;
                                                00B44 P.ADD:    .ASCII  <16>\Exception_Head!/\                                  ;
                                                00B53
                                                00B55 P.ADE:    .ASCII  <11>\Exception!/\                                       ;
                                                00B61 P.ADF:    .ASCII  <10>\Err_Halt!/\                                        ;
                                                00B6C P.ADG:    .ASCII  <9>\Summary!/\                                          ;
                                                00B76 P.ADH:    .ASCII  <15>\Program_Start!/\                                   ;
                                                00B85
                                                00B86 P.ADI:    .ASCII  <13>\Program_End!/\                                     ;
                                                00B94 P.ADJ:    .ASCII  <12>\First_Pass!/\                                      ;
                                                00BA1 P.ADK:    .ASCII  <10>\No_Tests!/\                                        ;
                                                00BAC P.ADL:    .ASCII  <12>\Abort_Test!/\                                      ;
                                                00BB9 P.ADM:    .ASCII  <15>\Abort_Program!/\                                   ;
                                                00BC8                                                                           ;
```

```
    2F  21  72  72  45  5F  64  6E  61  6D  6D  6F  43  0D  00BC9 P.ADN:   .ASCII  <13>\Command_Err!/\                          ;
    2F  21  74  75  4F  5F  64  6E  61  6D  6D  6F  43  0D  00BD7 P.ADO:   .ASCII  <13>\Command_Out!/\                          ;
2F  21  6F  66  6E  49  5F  6D  61  72  67  6F  72  50  0E  00BE5 P.ADP:   .ASCII  <14>\Program_Info!/\                         ;
            2F  21  72  72  45  5F  74  72  61  74  53  0B  00BF4 P.ADQ:   .ASCII  <11>\Start_Err!/\                           ;
72  6F  72  72  45  5F  65  63  6E  65  75  71  65  53  10  00C00 P.ADR:   .ASCII  <16>\Sequence_Error!/\                      ;
                                                    2F  21  00C0F                                                           ;
2F  21  74  73  65  54  6F  74  75  41  5F  44  52  43  0E  00C11 P.ADS:   .ASCII  <14>\CRD_AutoTest!/\                       ;
                2F  21  70  65  72  70  72  72  45  09  00C20 P.ADT:   .ASCII  <9>\Errprep!/\                               ;
    2F  21  72  6F  72  72  45  5F  6D  61  72  61  50  0D  00C2A P.ADU:   .ASCII  <13>\Param_Error!/\                        ;
            2F  21  74  72  61  74  53  5F  53  44  0A  00C38 P.ADV:   .ASCII  <10>\DS_Start!/\                            ;
        2F  21  66  6E  50  5F  74  70  69  72  63  53  0C  00C43 P.ADW:   .ASCII  <12>\Script_Pnf!/\                         ;
    2F  21  70  69  6B  53  5F  74  70  69  72  63  53  0D  00C50 P.ADX:   .ASCII  <13>\Script_Skip!/\                        ;
21  74  70  6D  6F  72  50  5F  74  70  69  72  63  53  0F  00C5E P.ADY:   .ASCII  <15>\Script_Prompt!/\                      ;
                                                    2F  00C6D                                                               ;
    2F  21  6F  68  63  45  5F  74  70  69  72  63  53  0D  00C6E P.ADZ:   .ASCII  <13>\Script_Echo!/\                        ;
2F  21  72  65  76  69  72  64  6F  4E  5F  6F  69  51  0E  00C7C P.AEA:   .ASCII  <14>\Qio_Nodriver!/\                      ;
2F  21  72  65  76  67  6E  6F  72  57  5F  6F  69  51  0E  00C8B P.AEB:   .ASCII  <14>\Qio_Wrongver!/\                      ;
        2F  21  70  64  61  76  6E  49  5F  6F  69  51  0C  00C9A P.AEC:   .ASCII  <12>\Qio_Invadp!/\                        ;
        2F  21  74  73  69  4C  5F  74  72  61  74  53  0C  00CA7 P.AED:   .ASCII  <12>\Start_List!/\                        ;
6F  63  65  70  79  74  20  6E  77  6F  6E  6B  6E  55  18  00CB4 P.AEE:   .ASCII  <24>\Unknown typecode = !UL!/\              ;
                2F  21  4C  55  21  20  3D  20  65  64  00CC3                                                               ;
65  70  79  74  20  64  65  6E  69  66  65  64  6E  55  1A  00CCD P.AEF:   .ASCII  <26>\Undefined typecode = !UL!/\            ;
            2F  21  4C  55  21  20  3D  20  65  64  6F  63  00CDC                                                           ;
```

```
                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                000C 00000      .ENTRY  DSR$PRINT_TYPECODE_NAME, Save R2,R3                 ; 1601
                                  53 00000000G   9F   9E 00002  MOVAB   @#DSX$PRINT, R3                                      ;
                                  52 00000000'   EF   9E 00009  MOVAB   P.AEF, R2                                           ;
                        25              00     04   AC   CF 00010  CASEL   TYPECODE, #0, #37                               ; 1635
0068            0065            005F            0059       00015 1$:  .WORD   2$-1$,-                                       ;
0083            007D            0077            0071       0001D      3$-1$,-                                       ;
009B            0095            008F            0089       00025      4$-1$,-                                       ;
00B3            00AD            00A7            00A1       0002D      5$-1$,-                                       ;
00CB            00C5            00BF            00B9       00035      6$-1$,-                                       ;
00E3            00DD            00D7            00D1       0003D      7$-1$,-                                       ;
00FB            00F5            00EF            00E9       00045      8$-1$,-                                       ;
0113            010D            0107            0101       0004D      9$-1$,-                                       ;
0127            0122            011D            0118       00055      10$-1$,-                                      ;
                                0131            012C       0005D      11$-1$,-                                      ;
                                                                      12$-1$,-                                      ;
                                                                      13$-1$,-                                      ;
                                                                      14$-1$,-                                      ;
                                                                      15$-1$,-                                      ;
                                                                      16$-1$,-                                      ;
                                                                      17$-1$,-                                      ;
                                                                      18$-1$,-                                      ;
                                                                      19$-1$,-                                      ;
                                                                      20$-1$,-                                      ;
                                                                      21$-1$,-                                      ;
                                                                      22$-1$,-                                      ;
                                                                      23$-1$,-                                      ;
                                                                      25$-1$,-                                      ;
                                                                      27$-1$,-                                      ;
                                                                      29$-1$,-                                      ;
```

```
                                                       31$-1$,-                                 ;
                                                       33$-1$,-                                 ;
                                                       35$-1$,-                                 ;
                                                       37$-1$,-                                 ;
                                                       39$-1$,-                                 ;
                                                       41$-1$,-                                 ;
                                                       43$-1$,-                                 ;
                                                       45$-1$,-                                 ;
                                                       46$-1$,-                                 ;
                                                       48$-1$,-                                 ;
                                                       50$-1$,-                                 ;
                                                       52$-1$,-                                 ;
                                                       54$-1$                                   ;
               04    AC   DD 00061           PUSHL     TYPECODE                                 ; 1680
                     52   DD 00064           PUSHL     R2                                       ;
                     01   DD 00066           PUSHL     #1                                       ;
                     03   DD 00068           PUSHL     #3                                       ;
         63           04  FB 0006A           CALLS     #4, DSX$PRINT                            ;
                          04 0006D           RET                                                ;
             FDFB    C2   9F 0006E 2$:       PUSHAB    P.ACS                                    ; 1637
                     7C   11 00072           BRB       24$                                      ;
             FE05    C2   9F 00074 3$:       PUSHAB    P.ACT                                    ; 1638
                     7C   11 00078           BRB       26$                                      ;
             FE11    C2   9F 0007A 4$:       PUSHAB    P.ACU                                    ; 1639
                     7C   11 0007E           BRB       28$                                      ;
             FE1F    C2   9F 00080 5$:       PUSHAB    P.ACV                                    ; 1640
                     7C   11 00084           BRB       30$                                      ;
             FE2F    C2   9F 00086 6$:       PUSHAB    P.ACW                                    ; 1641
                     7C   11 0008A           BRB       32$                                      ;
             FE38    C2   9F 0008C 7$:       PUSHAB    P.ACX                                    ; 1642
                     7C   11 00090           BRB       34$                                      ;
             FE41    C2   9F 00092 8$:       PUSHAB    P.ACY                                    ; 1643
                     7C   11 00096           BRB       36$                                      ;
             FE4B    C2   9F 00098 9$:       PUSHAB    P.ACZ                                    ; 1644
                     7C   11 0009C           BRB       38$                                      ;
             FE55    C2   9F 0009E 10$:      PUSHAB    P.ADA                                    ; 1645
                     7C   11 000A2           BRB       40$                                      ;
             FE5E    C2   9F 000A4 11$:      PUSHAB    P.ADB                                    ; 1646
                     7C   11 000A8           BRB       42$                                      ;
             FE6B    C2   9F 000AA 12$:      PUSHAB    P.ADC                                    ; 1647
                     7B   11 000AE           BRB       44$                                      ;
             FE77    C2   9F 000B0 13$:      PUSHAB    P.ADD                                    ; 1648
                     7F   11 000B4           BRB       47$                                      ;
             FE88    C2   9F 000B6 14$:      PUSHAB    P.ADE                                    ; 1649
                     7E   11 000BA           BRB       49$                                      ;
             FE94    C2   9F 000BC 15$:      PUSHAB    P.ADF                                    ; 1650
                     7D   11 000C0           BRB       51$                                      ;
             FE9F    C2   9F 000C2 16$:      PUSHAB    P.ADG                                    ; 1651
                     7C   11 000C6           BRB       53$                                      ;
             FEA9    C2   9F 000C8 17$:      PUSHAB    P.ADH                                    ; 1652
                     7B   11 000CC           BRB       55$                                      ;
             FEB9    C2   9F 000CE 18$:      PUSHAB    P.ADI                                    ; 1653
                     75   11 000D2           BRB       55$                                      ;
             FEC7    C2   9F 000D4 19$:      PUSHAB    P.ADJ                                    ; 1654
                     6F   11 000D8           BRB       55$                                      ;
             FED4    C2   9F 000DA 20$:      PUSHAB    P.ADK                                    ; 1655
                     69   11 000DE           BRB       55$                                      ;
```

```
                                FEDF    C2  9F 000E0 21$:    PUSHAB  P.ADL                           ; 1656
                                        63  11 000E4        BRB     55$                             ;
                                FEEC    C2  9F 000E6 22$:    PUSHAB  P.ADM                           ; 1657
                                        5D  11 000EA        BRB     55$                             ;
                                FEFC    C2  9F 000EC 23$:    PUSHAB  P.ADN                           ; 1658
                                        57  11 000F0 24$:    BRB     55$                             ;
                                FF0A    C2  9F 000F2 25$:    PUSHAB  P.ADO                           ; 1659
                                        51  11 000F6 26$:    BRB     55$                             ;
                                FF18    C2  9F 000F8 27$:    PUSHAB  P.ADP                           ; 1660
                                        4B  11 000FC 28$:    BRB     55$                             ;
                                FF27    C2  9F 000FE 29$:    PUSHAB  P.ADQ                           ; 1661
                                        45  11 00102 30$:    BRB     55$                             ;
                                FF33    C2  9F 00104 31$:    PUSHAB  P.ADR                           ; 1662
                                        3F  11 00108 32$:    BRB     55$                             ;
                                FF44    C2  9F 0010A 33$:    PUSHAB  P.ADS                           ; 1663
                                        39  11 0010E 34$:    BRB     55$                             ;
                                FF53    C2  9F 00110 35$:    PUSHAB  P.ADT                           ; 1664
                                        33  11 00114 36$:    BRB     55$                             ;
                                FF5D    C2  9F 00116 37$:    PUSHAB  P.ADU                           ; 1665
                                        2D  11 0011A 38$:    BRB     55$                             ;
                                FF6B    C2  9F 0011C 39$:    PUSHAB  P.ADV                           ; 1666
                                        27  11 00120 40$:    BRB     55$                             ;
                                FF76    C2  9F 00122 41$:    PUSHAB  P.ADW                           ; 1667
                                        21  11 00126 42$:    BRB     55$                             ;
                                  83    A2  9F 00128 43$:    PUSHAB  P.ADX                           ; 1668
                                        1C  11 0012B 44$:    BRB     55$                             ;
                                  91    A2  9F 0012D 45$:    PUSHAB  P.ADY                           ; 1669
                                        17  11 00130        BRB     55$                             ;
                                  A1    A2  9F 00132 46$:    PUSHAB  P.ADZ                           ; 1670
                                        12  11 00135 47$:    BRB     55$                             ;
                                  AF    A2  9F 00137 48$:    PUSHAB  P.AEA                           ; 1671
                                        0D  11 0013A 49$:    BRB     55$                             ;
                                  BE    A2  9F 0013C 50$:    PUSHAB  P.AEB                           ; 1672
                                        08  11 0013F 51$:    BRB     55$                             ;
                                  CD    A2  9F 00141 52$:    PUSHAB  P.AEC                           ; 1673
                                        03  11 00144 53$:    BRB     55$                             ;
                                  DA    A2  9F 00146 54$:    PUSHAB  P.AED                           ; 1674
                                        01  DD 00149 55$:    PUSHL   #1                              ;
                                        7E  D4 0014B         CLRL    -(SP)                           ;
                                    63      03  FB 0014D         CALLS   #3, DSX$PRINT               ;
                                            04  00150         RET                                     ; 1682

; Routine Size:  337 bytes,    Routine Base:  CODE + 0786

;   1683  1     END                                     ! End of CRDIMAGE module
;   1684  0     ELUDOM



;                                       PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
```

```
;  DATA                          3304  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                           132  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          2263  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                              Library Statistics
;
;                                      -------- Symbols --------      Pages       Processing
;       File                           Total    Loaded   Percent     Mapped      Time
;
;  DRB1:[DS.WORK]DS.L32;159              653        10         1         42       00:00.2
;  DRB1:[DS.WORK]DIAG.L32;265            784        15         1         85       00:00.2
;  DRB1:[DS.WORK]CRD.L32;265             483         9         1         62       00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7       18017        12         0        975       00:05.8



;                                    COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE CRDIMAGE

; Size:          2263 code + 3436 data bytes
; Run Time:         01:15.4
; Elapsed Time:     03:10.9
; Lines/CPU Min:     1340
; Lexemes/CPU-Min: 34270
; Memory Used:  455 pages
; Compilation Complete
```

able of contents

(1) 197 CONVERT BINARY TIME TO ASCII STRING
(1) 292 CONVERT ASCII STRING TO BINARY TIME
(1) 524 CONVERT BINARY TIME TO NUMERIC TIME

```
                        0000     .1          .TITLE  CVRTIM  *** CVRTIM time conversion routines
                        0000     .2          .IDENT  /V02-02/
-2                      0000      3
                        000C      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1976, 1977, 1978, 1979, 1980                               *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                        0000     11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER     *
                        0000     13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
                        0000     18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 6-JAN-76
                        0000     27 ;
                        0000     28 ; SYSTEM SERVICES TO CONVERT TIME
                        0000     29 ;
                        0000     30 ;       CONVERT BINARY TIME TO ASCII STRING
                        0000     31 ;       CONVERT ASCII STRING TO BINARY TIME
                        0000     32 ;       CONVERT BINARY TIME TO NUMERIC FORMAT
                        0000     33 ;
                        0000     34 ;       THE CONVERSION ALGORITHMS USED HEREIN WERE DEVELOPED BY P. CONKLIN,
                        0000     35 ;       M. SPIER, AND D. ROSENBERY ON THE PDP-10.
                        0000     36 ;
                        0000     37 ; MODIFIED BY:
                        0000     38 ;
                        0000     .1 ;                Roger Riggs, 8-May-1980, Version 5.04
                        0000     .2 ;       02       Emulate character string instructions
                        0000     .3 ;--
                        0000     39 ;       V02      LMK0001         LEN KAWELL      14-MAR-1979
                        0000     40 ;                FIX LOSS OF PRECISION IN HUNDRETHS OF SECONDS IN $NUMTIM
                        0000     41 ;
                        0000     42 ;
                        0000     43 ; MACRO LIBRARY CALLS
                        0000     44 ;
                        0000     45
                        0000     46          $SSDEF                              ;DEFINE SYSTEM STATUS VALUES
                        0000     47
                        0000     48 ;
                        0000     49 ; LOCAL SYMBOLS
                        0000     50 ;
                        0000     51 ; ARGUMENT LIST OFFSET DEFINTIONS FOR CONVERT BINARY TIME TO ASCII STRING
                        0000     52 ;
                        0000     53
00000004                0000     54 ATIMLEN=4                                   ;ADDRESS OF WORD TO STORE LENGTH
```

```
                     00000008  0000     55 ATIMBUF=8                                ;ADDRESS OF OUTPUT BUFFER DESCRIPTOR
                     0000000C  0000     56 ATIMADR=12                               ;ADDRESS OF 64-BIT ABSOLUTE OR DELTA TIME
                     00000010  0000     57 ACVTFLG=16                               ;CONVERSION INDICATOR
                               0000     58
                               0000     59 ;
                               0000     60 ; ARGUMENT LIST OFFSET DEFINITIONS FOR CONVERT ASCII STRING TO BINARY TIME
                               0000     61 ;
                               0000     62
                     00000004  0000     63 BTIMBUF=4                                ;ADDRESS OF ASCII STRING DESCRIPTOR
                     00000008  0000     64 BTIMADR=8                                ;ADDRESS TO STORE 64-BIT ABSOLUTE OR DELTA T
                               0000     65
                               0000     66 ;
                               0000     67 ; ARGUMENT LIST OFFSET DEFINITIONS FOR CONVERT BINARY TIME TO NUMERIC TIME
                               0000     68 ;
                               0000     69
                     00000004  0000     70 NTIMBUF=4                                ;ADDRESS OF 7-WORD BUFFER TO RECEIVE TIME
                     00000008  0000     71 NTIMADR=8                                ;ADDRESS OF 64-BIT ABSOLUTE OR DELTA TIME
                               0000     72
                               0000     73 ;
                               0000     74 ; CONVERSION CONSTANTS
                               0000     75 ;
                               0000     76 ; TOTAL DAYS IN A CENTURY
                               0000     77 ;
                               0000     78
                     00008EAC  0000     79 CENTURYDAYS=<100*365>+<100/4>-<100/100> ;
                               0000     80
                               0000     81 ;
                               0000     82 ; AVERAGE QUARTER DAYS PER CENTURY
                               0000     83 ;
                               0000     84
                     00023AB1  0000     85 QDAYSPCENT=<<<100*365>+<100/4>-<100/100>>*4>+<400/400> ;
                               0000     86
                               0000     87 ;
                               0000     88 ; AVERAGE QUARTER DAYS PER YEAR
                               0000     89 ;
                               0000     90
                     000005B5  0000     91 QDAYSPYEAR=<365*4>+1                     ;
                               0000     92
                               0000     93 ;
                               0000     94 ; TOTAL DAYS IN A QUADRICENTURY
                               0000     95 ;
                               0000     96
                     00023AB1  0000     97 QUADRIDAYS=<400*365>+<400/4>-<400/100>+<400/400> ;
                               0000     98
                               0000     99 ;
                               0000    100 ; TOTAL DAYS IN A QUADYEAR
                               0000    101 ;
                               0000    102
                     000005B5  0000    103 QUADYEARDAYS=<365*4>+1                   ;
                               0000    104
                               0000    105 ;
                               0000    106 ; OFFSET IN DAYS FROM 1-JAN-1501 TO 17-NOV-1858
                               0000    107 ;
                               0000    108
                     0001FE98  0000    109 TIMOFF1=<<1858-1501>*365>+<<1858-1501>/4>-<<1858-1501>/100>+<<1858-1501>/400>+ -;
                               0000    110         31+28+31+30+31+30+31+31+30+31+17 ;
                               0000    111
```

```
                                0000   112 ;
                                0000   113 ; OFFSET IN DAYS FROM 1-JAN-1601 TO 17-NOV-1858
                                0000   114 ;
                                0000   115
                                0000   116 TIMOFF2=<<1858-1601>*365>+<<1858-1601>/4>-<<1858-1601>/100>+<<1858-1601>/400>+ -;
                     00016FEC   0000   117         31+28+31+30+31+30+31+31+30+31+17 ;
                                0000   118
                                0000   119 ;
                                0000   120 ; CHARACTER CODE DEFINITIONS
                                0000   121 ;
                                0000   122
                     00000020   0000   123 BLANK=32                                          ;
                     0000003A   0000   124 COLON=58                                          ;
                     0000002D   0000   125 HYPHEN=45                                         ;
                     00000039   0000   126 NINE=57                                          ;
                     00000030   0000   127 ONE=48                                           ;
                     0000002E   0000   128 PERIOD=46                                        ;
                                0000   129
                                0000   130 ;
                                0000   131 ; NUMERIC TIME BUFFER OFFSET DEFINITIONS
                                0000   132 ;
                                0000   133
                     00000000   0000   134 YEAR=0                                           ;
                     00000002   0000   135 MONTH=2                                          ;
                     00000004   0000   136 DAY=4                                            ;
                     00000006   0000   137 HOUR=6                                           ;
                     00000008   0000   138 MINUTE=8                                        ;
                     0000000A   0000   139 SECOND=10                                       ;
                     0000000C   0000   140 HUNDREDTH=12                                    ;
                                0000   141
                                0000   142 ;
                                0000   143 ; LOCAL DATA
                                0000   144 ;
                                0000   145 ; MONTH, DAY CONVERSION TABLE
                                0000   146 ;
                           00000000     .1          .PSECT  SEP, SHR, EXE, WRT, LONG
-1                              0000   148 DATETABLE:                                       ;DATE CONVERSION TABLE
                             1F 0000   149          .BYTE   31                              ;JANUARY
                             1D 0001   150          .BYTE   29                              ;FEBRUARY
                             1F 0002   151          .BYTE   31                              ;MARCH
                             1E 0003   152          .BYTE   30                              ;APRIL
                             1F 0004   153          .BYTE   31                              ;MAY
                             1F 0005   154          .BYTE   30                              ;JUNE
                             1F 0006   155          .BYTE   31                              ;JULY
                             1F 0007   156          .BYTE   31                              ;AUGUST
                             1E 0008   157          .BYTE   30                              ;SEPTEMBER
                             1F 0009   158          .BYTE   31                              ;OCTOBER
                             1E 000A   159          .BYTE   30                              ;NOVEMBER
                             1F 0008   160          .BYTE   31                              ;DECEMBER
                                000C   161
                                000C   162 ;
                                000C   163 ; MONTH CONVERSION TABLE
                                000C   164 ;
                                000C   165
                                000C   166 MONTHTAB:                                       ;
                  4E 41 4A 03   000C   167          .ASCII  <3>/JAN/                        ;
                  42 45 46 03   0010   168          .ASCII  <3>/FEB/                        ;
```

```
                52 41 4D 03  0014   169          .ASCII  <3>/MAR/                ;
                52 50 41 03  0018   170          .ASCII  <3>/APR/                ;
                59 41 4D 03  001C   171          .ASCII  <3>/MAY/                ;
                4E 55 4A 03  0020   172          .ASCII  <3>/JUN/                ;
                4C 55 4A 03  0024   173          .ASCII  <3>/JUL/                ;
                47 55 41 03  0028   174          .ASCII  <3>/AUG/                ;
                50 45 53 03  002C   175          .ASCII  <3>/SEP/                ;
                54 43 4F 03  0030   176          .ASCII  <3>/OCT/                ;
                56 4F 4E 03  0034   177          .ASCII  <3>/NOV/                ;
                43 45 44 03  0038   178          .ASCII  <3>/DEC/                ;
                             003C   179
                             003C   180 ;
                             003C   181 ; HOURS, MINUTES, SECONDS, HUNDREDTHS CONVERSION TABLE
                             003C   182 ;
                             C03C   183
                             003C   184 TIMETABLE:                               ;TIME CONVERSION TABLE
                          64 003C   185          .BYTE   100                     ;HUNDREDTHS
                          3C 003D   186          .BYTE   60                      ;SECONDS
                          3C 003E   187          .BYTE   60                      ;MINUTES AND HOURS
                             003F   188
                             003F   189 ;
                             003F   190 ; CONVERSION CONTROL STRINGS
                             003F   191 ;
                             003F   192
          21 2D 57 53 32 21  003F   193 DATE.    .ASCII  /!2SW-!AC-!4ZW /        ;
          5A 34 21 2D 43 41  0045
                       20 57 004B
             20 57 53 34 21  004D   194 DELTA:   .ASCII  /!4SW /                 ;
       21 3A 57 5A 32 21     0052   195 TIME:    .ASCII  /!2ZW:!2ZW:!2ZW.!2ZW/   ;
       32 21 3A 57 5A 32     0058
       5A 32 21 2E 57 5A     005E
                    57 0064
```

```
                                0065   197          .SBTTL  CONVERT BINARY TIME TO ASCII STRING
                                0065   198 ;+
                                0065   199 ; EXE$ASCTIM - CONVERT BINARY TIME TO ASCII STRING
                                0065   200 ;
                                0065   201 ; THIS SERVICE PROVIDES THE CAPABILITY TO CONVERT AN ABSOLUTE OR DELTA
                                0065   202 ; TIME FROM 64-BIT FORMAT TO AN ASCII STRING.
                                0065   203 ;
                                0065   204 ; INPUTS:
                                0065   205 ;
                                0065   206 ;       ATIMLEN(AP) = ADDRESS OF WORD TO RECEIVE OUTPUT LENGTH.
                                0065   207 ;       ATIMBUF(AP) = ADDRESS OF OUTPUT BUFFER DESCRIPTOR.
                                0065   208 ;       ATIMADR(AP) = ADDRESS OF 64-BIT TIME VALUE. IF ZERO, THEN THE CURRENT
                                0065   209 ;                     SYSTEM TIME IS USED. POSITIVE VALUES ARE INTERPRETED AS
                                0065   210 ;                     ABSOLUTE TIMES AND NEGATIVE VALUES AS DELTA TIMES.
                                0065   211 ;       ACVTFLG(AP) = CONVERSION INDICATOR.
                                0065   212 ;                     LOW BIT CLEAR INDICATES BOTH DATE AND TIME ARE TO BE CON-
                                0065   213 ;                           VERTED.
                                0065   214 ;                     LOW BIT SET INDICATES ONLY TIME IS TO BE CONVERTED.
                                0065   215 ;
                                0065   216 ; OUTPUTS:
                                0065   217 ;
                                0065   218 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CONVERT TIME TO ASCII.
                                0065   219 ;
                                0065   220 ;                     R0 = SS$_ACCVIO - 64-BIT TIME VALUE OR OUTPUT BUFFER DESCRIPTOR
                                0065   221 ;                          CANNOT BE READ BY CALLING ACCESS MODE, OR OUTPUT BUFFER
                                0065   222 ;                          CANNOT BE WRITTEN BY CALLING ACCESS MODE.
                                0065   223 ;
                                0065   224 ;                     R0 = SS$_IVTIME - SPECIFIED DELTA TIME IS GREATER THAN 9999
                                0065   225 ;                          DAYS.
                                0065   226 ;
                                0065   227 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0065   228 ;
                                0065   229 ;                     R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0065   230 ;-
                                0065   231
                                0065   232 EXE$ASCTIM::                                ;CONVERT TIME TO ASCII
                         007C   0065   233          .WORD   ^M<R2,R3,R4,R5,R6>        ;ENTRY MASK
7E    08 BC    7D       0067   234          MOVQ    @ATIMBUF(AP),-(SP)        ;SAVE OUTPUT BUFFER DESCRIPTOR
   56    5E    D0       006B   235          MOVL    SP,R6                     ;SAVE ADDRESS OF OUTPUT BUFFER DESCRIPTOR
         7E    D4       006E   236          CLRL    -(SP)                     ;CLEAR SPACE FOR LENGTH FROM FAO
   55    5E    D0       0070   237          MOVL    SP,R5                     ;SAVE ADDRESS OF LENGTH
         52    D4       0073   238          CLRL    R2                        ;ASSUME ABSOLUTE TIME SPECIFIED
53    0C AC    D0       0075   239          MOVL    ATIMADR(AP),R3            ;GET ADDRESS OF 64-BIT TIME VALUE
         07    13       0079   240          BEQL    10$                       ;IF EQL NONE SPECIFIED
   50    63    7D       007B   241          MOVQ    (R3),R0                   ;GET 64-BIT TIME VALUE
         02    18       007E   242          BGEQ    10$                       ;IF GEQ ABSOLUTE TIME
         52    D6       0080   243          INCL    R2                        ;INDICATE DELTA TIME
   5E    10    C2       0082   244 10$:     SUBL    #<<<7*2>+3>/4>*4,SP       ;ALLOCATE NUMERIC TIME BUFFER
   54    5E    D0       0085   245          MOVL    SP,R4                     ;SAVE ADDRESS OF NUMERIC TIME BUFFER
                        0088   246          $NUMTIM_S (R4),(R3)               ;CONVERT TIME TO NUMERIC FORMAT
      6F 50    E9       0093   247          BLBC    R0,60$                    ;IF LBC CONVERSION FAILURE
                        0096   248
                        0096   249 ;
                        0096   250 ; CONVERT TIME TO ASCII FORMAT
                        0096   251 ;
                        0096   252
   3E 10 AC    E8       0096   253          BLBS    ACVTFLG(AP),40$           ;IF LBS ONLY TIME IS TO BE CONVERTED
```

```
                12 52    E8  009A   254          BLBS    R2,20$                  ;IF LBS DELTA TIME SPECIFIED
                            009D   255
                            009D   256 ;
                            009D   257 ; CONVERT DATE
                            009D   258 ;
                            009D   259
          52    02 A4    3C  009D   260          MOVZWL  MONTH(R4),R2            ;GET NUMERIC MONTH VALUE
      52   FF62 CF42    DE  00A1   261          MOVAL   W^MONTHTAB-4[R2],R2     ;GET ADDRESS OF MONTH COUNTED STRING
           FF94 CF      DF  00A7   262          PUSHAL  W^DATE                  ;BUILD DESCRIPTOR FOR CONTROL STRING
                0E      DD  00AB   263          PUSHL   #DELTA-DATE             ;
                06      11  00AD   264          BRB     30$                     ;
                            00AF   265
                            00AF   266 ;
                            00AF   267 ; CONVERT DELTA TIME
                            00AF   268 ;
                            00AF   269
           FF9A CF      DF  00AF   270 20$:     PUSHAL  W^DELTA                 ;BUILD CONTROL STRING DESCRIPTOR
                05      DD  00B3   271          PUSHL   #TIME-DELTA             ;
           51   5E      D0  00B5   272 30$:     MOVL    SP,R1                   ;COPY ADDRESS OF CONTROL STRING DESCRIPTOR
                            00B8   273          $FAO_S  (R1),(R5),(R6),DAY(R4),R2,YEAR(R4) ;CONVERT DELTA TIME OR DATE
             36 50      E9  00CC   274          BLBC    R0,60$                  ;IF LBC CONVERT FAILURE
           66   65      A2  00CF   275          SUBW    (R5),(R6)               ;ANY SPACE LEFT IN TIME BUFFER?
                27      15  00D2   276          BLEQ    50$                     ;IF LEQ NO
      04 A6     65      C0  00D4   277          ADDL    (R5),4(R6)              ;UPDATE TIME BUFFER ADDRESS
                            00D8   278
                            00D8   279 ;
                            00D8   280 ; CONVERT TIME
                            00D8   281 ;
                            00D8   282
           FF76 CF      DF  00D8   283 40$:     PUSHAL  W^TIME                  ;BUILD CONTROL STRING DESCRIPTOR
                13      DD  00DC   284          PUSHL   #EXE$ASCTIM-TIME        ;
           51   5E      D0  00DE   285          MOVL    SP,R1                   ;COPY ADDRESS OF CONTROL STRING DESCRIPTOR
                            00E1   286          $FAO_S  (R1),2(R5),(R6),HOUR(R4),MINUTE(R4),SECOND(R4),HUNDREDTH(R4) ;
     51    04 AC      D0  00FB   287 50$:     MOVL    ATIMLEN(AP),R1          ;LENGTH ADDRESS SPECIFIED?
                04      13  00FF   288          BEQL    60$                     ;IF EQL NO
61   65    85          A1  0101   289          ADDW3   (R5)+,(R5),(R1)         ;COMPUTE AND RETURN OUTPUT LENGTH
                        04  0105   290 60$:     RET                             ;
```

```
                                0106   292          .SBTTL  CONVERT ASCII STRING TO BINARY TIME
                                0106   293 ;+
                                0106   294 ; EXE$BINTIM - CONVERT ASCII STRING TO BINARY TIME
                                0106   295 ;
                                0106   296 ; THIS SERVICE PROVIDES THE CAPABILITY TO CONVERT AN ASCII STRING TO A
                                0106   297 ; 64-BIT ABSOLUTE OR DELTA TIME.
                                0106   298 ;
                                0106   299 ; INPUTS:
                                0106   300 ;
                                0106   301 ;       BTIMBUF(AP) = ADDRESS OF ASCII STRING DESCRIPTOR.
                                0106   302 ;       BTIMADR(AP) = ADDRESS TO STORE 64-BIT TIME VALUE.
                                0106   303 ;
                                0106   304 ; OUTPUTS:
                                0106   305 ;
                                0106   306 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CONVERT TIME TO ASCII.
                                0106   307 ;
                                0106   308 ;               R0 = SS$_IVTIME - ASCII STRING HAS INVALID SYNTAX OR TIME
                                0106   309 ;                    COMPONENT IS OUT OF RANGE.
                                0106   310 ;
                                0106   311 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0106   312 ;
                                0106   313 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0106   314 ;-
                                0106   315
                                0106   316 EXE$BINTIM::                                ;CONVERT ASCII STRING TO BINARY TIME
                          01FC  0106   317          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8> ;ENTRY MASK
                     5E   10    C2  0108   318          SUBL    #<<<7*2>+3>/4>*4,SP      ;ALLOCATE NUMERIC TIME BUFFER
                     57   5E    D0  010B   319          MOVL    SP,R7                    ;SAVE ADDRESS OF NUMERIC TIME BUFFER
                55   04 BC    7D  010E   320          MOVQ    @BTIMBUF(AP),R5          ;GET ADDRESS AND LENGTH OF ASCII STRING
                        58    D4  0112   321          CLRL    R8                       ;ASSUME DELTA TIME
                        55    B7  0114   322 10$:     DECW    R5                       ;ANY MORE CHARACTERS?
                        6F    19  0116   323          BLSS    30$                      ;IF LSS NO
                86      20    91  0118   324          CMPB    #BLANK,(R6)+             ;SKIP LEADING BLANK?
                        F7    13  011B   325          BEQL    10$                      ;IF EQL YES
                        55    B6  011D   326          INCW    R5                       ;CORRECT NUMBER OF CHARACTERS
                              011F    .1 ;+
                              011F    .2 ;       REPLACED        LOCC    #HYPHEN,R5,-(R6)        ;ABSOLUTE TIME FORMAT?
                              011F    .3 ;-
                51    76    9E  011F    .4          MOVAB   -(R6),R1
                50    55    3C  0122    .5          MOVZWL  R5,R0
                        0C    13  0125    .6          BEQL    10002$
                              0127    .7 10000$:
                2D    81    91  0127    .8          CMPB    (R1)+,#HYPHEN
                        05    13  012A    .9          BEQL    10001$
                   F8   50    F5  012C   .10          SOBGTR  R0,10000$
                        02    11  012F   .11          BRB     10002$
                              0131   .12 10001$:
                        51    D7  0131   .13          DECL    R1
                              0133   .14 10002$:
-1                      52    13  0133   328          BEQL    30$                      ;IF EQL NO
                        58    D6  0135   329          INCL    R8                       ;INDICATE ABSOLUTE TIME
                              0137   330          $NUMTIM_S (R7)                   ;CONVERT CURRENT TIME TO NUMERIC FORMAT
                              0142   331
                              0142   332 ;
                              0142   333 ; CONVERT ABSOLUTE TIME
                              0142   334 ;
                              0142   335
```

```
                54    04 A7    DE  0142   336          MOVAL   DAY(R7),R4              ;SET ADDRESS TO STORE DAY
                         6B    10  0146   337          BSBB    CONVERT                 ;CONVERT DAY FIELD
                         2D    2D  0148   338          .BYTE   HYPHEN                  ;EXPECTED TERMINATOR
                         55    B5  0149   339          TSTW    R5                      ;ANY MORE CHARACTERS?
                         64    13  014B   340          BEQL    60$                     ;IF EQL NO
                      86 2D    91  014D   341          CMPB    #HYPHEN,(R6)+           ;MONTH FIELD VOID?
                         2E    13  0150   342          BEQL    20$                     ;IF EQL YES
                               0152    .1 ;+
                               0152    .2 ;       REPLACED        MATCHC  #3,-(R6),#4*12,W^MONTHTAB ;SEARCH FOR MONTH SUBSTRIN
                               0152    .3 ;-
                      51       76  9E  0152    .4          MOVAB   -(R6),R1
                   50 61       08  78  0155    .5          ASHL    #8,(R1),R0
                      50       03  90  0159    .6          MOVB    #3,R0
                               52  D4  015C    .7          CLRL    R2                      ; Clear month index
                               C15E    .8
                FEA8 CF42      50  D1  015E    .9 15$:     CMPL    R0,MONTHTAB[R2]         ; Correct month?
                               06  13  0164   .10          BEQL    19$                     ; Branch if so
                      F4 52    0C  F2  0166   .11          AOBLSS  #12,R2,15$              ; Increment month and branch if more
                               74  11  016A   .12          BRB     IVTIME                  ; Month not found, flag invalid
                               016C   .13
                      52       01  A1  016C   .14 19$:     ADDW3   #1,R2,MONTH(R7)         ;CONVERT TO MONTH AND STORE
                         02 A7         016F
-6                    56       03  C0  0171   349          ADDL    #3,R6                   ;UPDATE ADDRESS OF ASCII STRING
                      55       03  A2  0174   350          SUBW    #3,R5                   ;UPDATE COUNT OF REMAINING CHARACTERS
                               67  19  0177   351          BLSS    IVTIME                  ;IF LSS INVALID SYNTAX
                               36  13  0179   352          BEQL    60$                     ;IF EQL END OF STRING
                      86       2D  91  017B   353          CMPB    #HYPHEN,(R6)+           ;FIELD TERMINATED PROPERLY?
                               60  12  017E   354          BNEQ    IVTIME                  ;IF NEQ NO
                               55  B7  0180   355 20$:      DECW    R5                      ;DECREMENT COUNT OF REMAINING CHARACTERS
                54             67  DE  0182   356          MOVAL   YEAR(R7),R4             ;SET ADDRESS TO STORE YEAR
                               0A  11  0185   357          BRB     40$                     ;
                                       0187   358
                                       0187   359 ;
                                       0187   360 ; CONVERT DELTA TIME
                                       0187   361 ;
                                       0187   362
                54             67  DE  0187   363 30$:      MOVAL   YEAR(R7),R4             ;GET ADDRESS TO STORE YEAR
                               84  D4  018A   364          CLRL    (R4)+                   ;CLEAR YEAR AND MONTH
                               64  7C  018C   365          CLRQ    (R4)                    ;CLEAR DAY, HOUR, MINUTE, AND SECOND
                            0C A7  B4  018E   366          CLRW    HUNDREDTH(R7)           ;CLEAR HUNDREDTH
                               20  10  0191   367 40$:      BSBB    CONVERT                 ;CONVERT RELATIVE DAY OR YEAR FIELD
                               20  0193   368          .BYTE   BLANK                   ;EXPECTED TERMINATOR
                               55  B7  0194   369 50$:      DECW    R5                      ;ANY REMAINING CHARACTERS?
                               19  19  0196   370          BLSS    60$                     ;IF LSS NO
                86             20  91  0198   371          CMPB    #BLANK,(R6)+            ;NEXT CHARACTER BLANK?
                               F7  13  019B   372          BEQL    50$                     ;IF EQL YES
                               56  D7  019D   373          DECL    R6                      ;BACK UP TO NONBLANK CHARACTER
                               55  D6  019F   374          INCL    R5                      ;ADJUST REMAINING CHARACTER COUNT
                                       01A1   375
                                       01A1   376 ;
                                       01A1   377 ; CONVERT TIME
                                       01A1   378 ;
                                       01A1   379
            54    06 A7    DE  01A1   380          MOVAL   HOUR(R7),R4             ;SET ADDRESS TO STORE HOUR
                      0C    10  01A5   381          BSBB    CONVERT                 ;CONVERT HOUR FIELD
                            3A  01A7   382          .BYTE   COLON                   ;EXPECTED TERMINATOR
                            09  10  01A8   383          BSBB    CONVERT                 ;CONVERT MINUTE FIELD
```

```
                     3A  01AA   384          .BYTE   COLON                   ;EXPECTED TERMINATOR
                 06  10  01AB   385          BSBB    CONVERT                 ;CONVERT SECOND FIELD
                     2E  01AD   386          .BYTE   PERIOD                  ;EXPECTED TERMINATOR
                 03  10  01AE   387          BSBB    CONVERT                 ;CONVERT HUNDREDTH FIELD
                     20  01B0   388          .BYTE   BLANK                   ;EXPECTED TERMINATOR
                 33  11  01B1   389 60$:     BRB     CVRTIME                 ;
                         01B3   390
                         01B3   391 ;
                         01B3   392 ; SUBROUTINE TO CONVERT NUMERIC FIELD TO BINARY
                         01B3   393 ;
                         01B3   394
                         01B3   395 CONVERT:                              ;CONVERT FIELD
                 50  D4  01B3   396          CLRL    R0                      ;CLEAR ACCUMULATED VALUE
                 84  B5  01B5   397 10$:     TSTW    (R4)+                   ;POINT PAST NEXT FIELD
                 55  B7  01B7   398          DECW    R5                      ;ANY MORE CHARACTERS?
                 2B  19  01B9   399          BLSS    CVRTIME                 ;IF LSS NO
             51  86  9A  01BB   400          MOVZBL  (R6)+,R1                ;GET NEXT CHARACTER
          00 BE  51  91  01BE   401          CMPB    R1,@(SP)                ;EXPECTED TERMINATOR?
                 19  13  01C2   402          BEQL    20$                     ;IF EQL YES
             51  30  C2  01C4   403          SUBL    #ONE,R1                 ;SUBTRACT OUT CHARACTER BIAS
                 17  19  01C7   404          BLSS    IVTIME                  ;IF LSS INVALID CHARACTER
             51  09  D1  01C9   405          CMPL    #NINE-ONE,R1            ;RESULT VALUE WITHIN RANGE?
                 12  19  01CC   406          BLSS    IVTIME                  ;IF LSS INVALID CHARACTER
             50  0A  A4  01CE   407          MULW    #10,R0                  ;MULTIPLY PARTIAL RESULT BY 10
                 0D  1D  01D1   408          BVS     IVTIME                  ;IF VS INVALID TIME
             50  51  A0  01D3   409          ADDW    R1,R0                   ;ACCUMULATE VALUE
                 08  1D  01D6   410          BVS     IVTIME                  ;IF VS INVALID TIME VALUE
                 74  50  B0  01D8   411          MOVW    R0,-(R4)            ;STORE VALUE
                 D8  11  01DB   412          BRB     10$                     ;
                 6E  D6  01DD   413 20$:     INCL    (SP)                    ;INCREMENT PAST TERMINATOR
                     05  01DF   414          RSB                             ;
                         01E0   415
                         01E0   416 ;
                         01E0   417 ; INVALID SYNTAX OR TIME COMPONENT
                         01E0   418 ;
                         01E0   419
          50   0184  8F  3C  01E0   420 IVTIME: MOVZWL  #SS$_IVTIME,R0        ;SET INVALID TIME
                     04  01E5   421          RET                             ;
                         01E6   422
                         01E6   423 ;
                         01E6   424 ; CHECK CONVERTED DATE AND TIME VALUES
                         01E6   425 ;
                         01E6   426
                         01E6   427 CVRTIME:                              ;
              270F  8F  B1  01E6   428          CMPW    #9999,DAY(R7)           ;DAY WITHIN UPPER LIMIT?
                04  A7          01EA
                 F2  1F  01EC   429          BLSSU   IVTIME                  ;IF LSSU NO
           06 A7     18  B1  01EE   430          CMPW    #24,HOUR(R7)            ;HOUR WITHIN LIMITS?
                 EC  18  01F2   431          BLEQU   IVTIME                  ;IF LEQU NO
           08 A7     3C  B1  01F4   432          CMPW    #60,MINUTE(R7)          ;MINUTE WITHIN LIMITS?
                 E6  18  01F8   433          BLEQU   IVTIME                  ;IF LEQU NO
           0A A7     3C  B1  01FA   434          CMPW    #60,SECOND(R7)          ;SECOND WITHIN LIMITS?
                 E0  18  01FE   435          BLEQU   IVTIME                  ;IF LEQU NO
              0064  8F  B1  0200   436          CMPW    #100,HUNDREDTH(R7)      ;HUNDREDTH WITHIN LIMITS?
                0C  A7          0204
                 D8  18  0206   437          BLEQU   IVTIME                  ;IF LEQU NO
          55   04  A7  3C  0208   438          MOVZWL  DAY(R7),R5              ;GET DAY VALUE
```

```
                03 58    E8  020C   439          BLBS    R8,5$                   ;IF LBS ABSOLUTE TIME
                 0097    31  020F   440          BRW     40$                     ;
                         0212   441
                         0212   442 ;
                         0212   443 ; CONVERT YEARS TO QUADRICENTURIES, CENTURIES, QUADYEARS, YEARS
                         0212   444 ;
                         0212   445
                    CC   13  0212   446 5$:      BEQL    IVTIME                  ;IF EQL INVALID TIME
                50  67   3C  0214   447          MOVZWL  YEAR(R7),R0             ;GET YEAR VALUE
           50   F9BF CO   3E  0217   448          MOVAW   -1601(R0),R0            ;CALCULATE YEARS PAST 1601
                    C2   19  021C   449          BLSS    IVTIME                  ;IF LSS INVALID TIME
                    51   D4  021E   450          CLRL    R1                      ;CLEAR HIGH PART OF DIVIDEND
            00000190 8F   7B  0220   451          EDIV    #400,R0,R0,R1           ;CALCULATE QUADRICENTURIES
           51   50   50        0226
                    52   D4  C229   452          CLRL    R2                      ;CLEAR HIGH PART OF DIVIDEND
            00000064 8F   7B  022B   453          EDIV    #100,R1,R1,R2           ;CALCULATE CENTURIES
           52   51   51        0231
                    53   D4  0234   454          CLRL    R3                      ;CLEAR HIGH PART OF DIVIDEND
           52   52   04   7B  0236   455          EDIV    #4,R2,R2,R3             ;CALCULATE QUADYEARS AND YEARS
                    53        023A
                         023B   456
                         023B   457 ;
                         023B   458 ; CONVERT QUADRICENTURIES, CENTURIES, QUADYEARS, YEARS TO DAYS
                         023B   459 ;
                         023B   460
           53   016D 8F   A4  023B   461          MULW    #365,R3                 ;CALCULATE NUMBER OF DAYS PAST LEAP YEAR
            000005B5 8F   7A  0240   462          EMUL    #QUADYEARDAYS,R2,R3,R2  ;CALCULATE NUMBER OF QUADYEAR DAYS AND SUM
           52   53   52        0246
            00008EAC 8F   C4  0249   463          MULL    #CENTURYDAYS,R1         ;CALCULATE NUMBER OF CENTURY DAYS
                    51        024F
            00023AB1 8F   7A  0250   464          EMUL    #QUADRIDAYS,R0,R1,R5    ;CALCULATE NUMBER OF QUADRIDAYS AND SUM
           55   51   50        0256
                    50   D4  0259   465          CLRL    R0                      ;CLEAR INITIAL LOOP INDEX
             56   02  A7   3C  025B   466          MOVZWL  MONTH(R7),R6            ;GET SPECIFIED MONTH VALUE
                55   52   C0  025F   467 10$:     ADDL    R2,R5                   ;ACCUMULATE TOTAL DAYS
       52   FD99 CF40  9A  0262   468          MOVZBL  W^DATETABLE[R0],R2      ;GET NUMBER OF DAYS IN MONTH
                50   01   D1  0268   469          CMPL    #1,R0                   ;SECOND MONTH OF YEAR?
                     1E   12  026B   470          BNEQ    30$                     ;IF NEQ NO
                53   67   3C  026D   471          MOVZWL  YEAR(R7),R3             ;GET SPECIFIED YEAR VALUE
                53   03   D3  0270   472          BITL    #3,R3                   ;YEAR MULTIPLE OF 4?
                     14   12  0273   473          BNEQ    20$                     ;IF NEQ NO
                     54   D4  0275   474          CLRL    R4                      ;CLEAR HIGH PART OF DIVIDEND
            00000064 8F   7B  0277   475          EDIV    #100,R3,R3,R4           ;CALCULATE CENTURY AND YEAR IN CENTURY
           54   53   53        027D
                     54   D5  0280   476          TSTL    R4                      ;YEAR MULTIPLE OF 100?
                     07   12  0282   477          BNEQ    30$                     ;IF NEQ NO
                53   03   D3  0284   478          BITL    #3,R3                   ;YEAR MULTIPLE OF 400?
                     02   13  0287   479          BEQL    30$                     ;IF EQL YES
                     52   D7  0289   480 20$:     DECL    R2                      ;REDUCE NUMBER OF DAYS IN MONTH
             D0  50   56   F2  028B   481 30$:     AOBLSS  R6,R0,10$               ;ANY MORE DAYS TO ACCUMULATE?
           50   0184 8F   3C  028F   482          MOVZWL  #SS*,/TIME,R0           ;ASSUME INVALID DAY OF MONTH
             51   04   A7   3C  0294   483          MOVZWL  DAY(R7),R1              ;GET SPECIFIED DAY
            00016FEC 8F   C2  0298   484          SUBL    #TIMOFF2,R5             ;SUBTRACT OUT NUMBER OF DAYS TO 17-NOV-1858
                     55        029E
                55   51   C0  029F   485          ADDL    R1,R5                   ;CALCULATE TOTAL NUMBER OF DAYS
                     57   19  02A2   486          BLSS    60$                     ;IF LSS INVALID TIME
                52   51   D1  02A4   487          CMPL    R1,R2                   ;DAY WITHIN LIMITS?
```

```
                    52   1A   02A7   488          BGTRU   60$                     ;IF GTRU NO
                                02A9   489
                                02A9   490 ;
                                02A9   491 ; CONVERT TIME TO TENTHS OF MICROSECONDS
                                02A9   492 ;
                                02A9   493
      50    06 A7   3C   02A9   494 50$:    MOVZWL  HOUR(R7),R0             ;GET HOUR VALUE
      51    08 A7   3C   02AD   495          MOVZWL  MINUTE(R7),R1           ;GET MINUTE VALUE
   51    50    3C   7A   02B1   496          EMUL    #60,R0,R1,R0            ;CONVERT HOURS TO MINUTES AND SUM
                   50   02B5
      51    0A A7   3C   02B6   497          MOVZWL  SECOND(R7),R1           ;GET SECOND VALUE
   51    50    3C   7A   02BA   498          EMUL    #60,R0,R1,R0            ;CONVERT MINUTES TO SECONDS AND SUM
                   50   02BE
      51    0C A7   3C   02BF   499          MOVZWL  HUNDREDTH(R7),R1        ;GET HUNDREDTH VALUE
    00000064 8F   7A   C2C3   500          EMUL    #100,R0,R1,R0           ;CONVERT SECONDS TO HUNDREDTHS AND SUM
   50    51    50   02C9
    000186A0 8F   7A   02CC   501          EMUL    #100000,R0,#0,R0        ;CONVERT TO TENTHS OF MICROSECONDS
   50    00    50   02D2
                                02D5   502
                                02D5   503 ;
                                02D5   504 ; CONVERT DAYS TO TENTHS OF MICROSECONDS
                                02D5   505 ;
                                02D5   506
    324A9A70 8F   7A   02D5   507          EMUL    #843750000,R5,#0,R2     ;MULTIPLY BY 864000000000/1024
   52    00    55   02DB
   52    52    0A   79   02DE   508          ASHQ    #10,R2,R2               ;MULTIPLY BY 1024
                                02E2   509
                                02E2   510 ;
                                02E2   511 ; COMBINE RESULTS AND STORE 64-BIT TIME
                                02E2   512 ;
                                02E2   513
      52    50   C0   02E2   514          ADDL    R0,R2                   ;ADD LOW ORDER PARTS
      53    51   D8   02E5   515          ADWC    R1,R3                   ;ADD HIGH ORDER PARTS
      50    01   3C   02E8   516          MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION
         09 58   E8   02EB   517          BLBS    R8,50$                  ;IF LBS ABSOLUTE TIME
      53    53   CE   02EE   518          MNEGL   R3,R3                   ;CONVERT TO DELTA TIME
      52    52   CE   02F1   519          MNEGL   R2,R2                   ;
      53    00   D9   02F4   520          SBWC    #0,R3                   ;
08 BC    52   7D   02F7   521 50$:    MOVQ    R2,@BTIMADR(AP)         ;STORE 64-BIT TIME VALUE
                   04   02FB   522 60$:    RET                             ;
```

B 1 INTERVAL TIMER ROUTINES.
C 1 INTERVAL TIMER ROUTINES.
D 1 INTERVAL TIMER ROUTINES.
E 1 INTERVAL TIMER ROUTINES.
F 1 Macros and Equated Symbols
G 1 Macros and Equated Symbols
H 1 Macros and Equated Symbols
I 1 Macros and Equated Symbols
J 1 Data Psect Declarations
K 1 DS$CanWait - CANCEL DELAY ROUT
L 1 $WAKE - Wakeup from hibernate
M 1 DSX$WaitMS - MILLISECOND DELAY
N 1 DSX$WaitMS - MILLISECOND DELAY
B 2 DSX$WaitUC - MICROSECOND DELAY
C 2 DSX$WaitUC - MICROSECOND DELAY
D 2 EXE$HIBER - HIBERNATE SYSTEM S
E 2 EXE$HIBER - HIBERNATE SYSTEM S
F 2 EXE$CANTIM - CANCEL TIMER SYST
G 2 EXE$CANTIM - CANCEL TIMER SYST
H 2 DSR$SETIMR_INI - SET TIMER QUE
I 2 DSR$SETIMR_INI - SET TIMER QUE
J 2 EXE$Setimr - SET TIMER SYSTEM
K 2 EXE$Setimr - SET TIMER SYSTEM
L 2 DSI$TimSrv - INTERVAL TIMER IN
M 2 DSI$TimSrv - INTERVAL TIMER IN
N 2 DSI$TIMER - Level 7 timer inte
B 3 DSI$TIMER - Level 7 timer inte
C 3 DSI$TIMER - Level 7 timer inte
D 3 DSI$TIMER - Level 7 timer inte
E 3 CLK$RE_INIT - SYSTEM TIMER INI
F 3 CLK$RE_INIT - SYSTEM TIMER INI
G 3 Symbol table
H 3 Symbol table
I 3 Symbol table
J 3 Symbol table
K 3 Psect synopsis
L 3 Cross reference
M 3 Cross reference
N 3 Cross reference
B 4 Cross reference
C 4 Cross reference
D 4 Cross reference
E 4 Cross reference
F 4 *** COMDRVSUB driver support r
G 4 *** COMDRVSUB driver support r
H 4 *** COMDRVSUB driver support r
I 4 COM$DELATTNAST - DELIVER ATTEN
J 4 COM$FLUSHATTNS - FLUSH ATTENTI
K 4 COM$POST - POST I.O COMPLETION
L 4 COM$DRVDEALMEM - DEALLOCATE DR
M 4 COM$SETATTNAST - SET UP ATTENT
N 4 COM$SETATTNAST - SET UP ATTENT
B 5 Symbol table
C 5 Symbol table
D 5 Psect synopsis
E 5 Cross reference
F 5 Cross reference
G 5 Cross reference
H 5 *** CONFIG Configure load devi
I 5 *** CONFIG Configure load devi
J 5 *** CONFIG Configure load devi
K 5 *** CONFIG Configure load devi
L 5 *** CONFIG Configure load devi
M 5 *** CONFIG Configure load devi
N 5 *** CONFIG Configure load devi
B 6 INI$LOAD_DEVICE Make ptables
C 6 INI$LOAD_DEVICE Make ptables
D 6 INI$LOAD_DEVICE Make ptables
E 6 DSP$CONFIG_LOAD Configure load
F 6 DSP$CONFIG_LOAD Configure load
G 6 DSP$CONFIG_LOAD Configure load
H 6 DSP$CONFIG_LOAD Configure load
I 6 DSP$CONFIG_LOAD Configure load
J 6 DSP$CONFIG_LOAD Configure load
K 6 DSP$CONFIG_LOAD Configure load
L 6 Common routines
M 6 Common routines
N 6 Common routines
B 7 configure RB730
C 7 configure RB730
D 7 configure RB730
E 7 Attach UDA-50/LESI load path
F 7 Attach UDA-50/LESI load path
G 7 Attach UDA-50/LESI load path
H 7 Symbol table
I 7 Symbol table
J 7 Symbol table
K 7 Psect synopsis
L 7 Cross reference
M 7 Cross reference
N 7 Cross reference
B 8 Cross reference
C 8 Cross reference
D 8 Cross reference
E 8 Cross reference
F 8 Cross reference
G 8 *** CONSOLE Console I/O handle
H 8 *** CONSOLE Console I/O handle
I 8 *** CONSOLE Console I/O handle
J 8 *** CONSOLE Console I/O handle
K 8 *** CONSOLE Console I/O handle
L 8 Libraries, Equated Symbols, Ma
M 8 Work Psect Declarations : [
N 8 Data Psect Declarations : [
B 9 Data Psect Declarations : [
C 9 Data Psect Declarations : [
D 9 Data Psect Declarations : [
E 9 VRSetWidth Routine - Set termi
F 9 VRSetWidth Routine - Set termi
G 9 VRShowWidth Routine - Show ter
H 9 VRShowWidth Routine - Show ter
I 9 KB_Poll Routine - Keyboard rea
J 9 KB_Poll Routine - Keyboard rea
K 9 RInput Routine - Flag driven t
L 9 RInput Routine - Flag driven t
M 9 RInput Routine - Flag driven t
N 9 RType Routine - Retype input l
B 10 RType Routine - Retype input l
C 10 Out Routines - Flag driven ter
D 10 Out Routines - Flag driven ter
E 10 Out Routines - Flag driven ter
F 10 Out Routines - Flag driven ter
G 10 WriteVBlk Routine - Console wr
H 10 WriteVBlk Routine - Console wr
I 10 ReadVBlk Routine - Console rea
J 10 ReadVBlk Routine - Console rea
K 10 ReadVBlk Routine - Console rea
L 10 ReadVBlk Routine - Console rea
M 10 SIZE_TERM Routine to size the
N 10 SIZE_TERM Routine to size the
B 11 SIZE_TERM Routine to size the
C 11 Symbol table
D 11 Symbol table
E 11 Symbol table
F 11 Psect synopsis
G 11 Cross reference
H 11 Cross reference
I 11 Cross reference
J 11 Cross reference
K 11 Cross reference
L 11 Cross reference
M 11 Cross reference
N 11 Cross reference
B 12 *** Loading and Unloading the
C 12 *** Loading and Unloading the
D 12 Libraries
E 12 Other Literals
F 12 External Routines
G 12 Psect Definitions
H 12 Local ASCIC Bindings
I 12 Own Storage
J 12 Own Storage
K 12 DSR$Load_CRD Routine
L 12 DSR$Load_CRD Routine
M 12 DSR$Load_CRD Routine
N 12 DSR$Load_CRD Routine
B 13 DSR$Load_CRD Routine
C 13 DSR$Load_CRD Routine
D 13 DSR$Load_CRD Routine
E 13 DSR$Load_CRD Routine
F 13 DSR$Load_CRD Routine
G 13 DSR$Load_CRD Routine
H 13 DSR$Load_CRD Routine
I 13 DSR$Unload_CRD Routine
J 13 DSR$Unload_CRD Routine
K 13 DSR$Unload_CRD Routine
L 13 DSR$Unload_CRD Routine
M 13 DSR$Unload_CRD Routine
N 13 DSR$Restore_CRD_Vectors Routin
B 14 DSR$Restore_CRD_Vectors Routin
C 14 Exchange_Dispatch_Vectors Rout
D 14 Exchange_Dispatch_Vectors Rout
E 14 Exchange_Dispatch_Vectors Rout
F 14 Exchange_Dispatch_Vectors Rout
G 14 CRD_Exit Routine
H 14 CRD_Exit Routine
I 14 CRD_Error Routine
J 14 CRD_Error Routine
K 14 CRD_Error Routine
L 14 CRD_Error Routine

```
M 14  CRD_Error Routine
N 14  Internal_Error Routine
B 15  Internal_Error Routine
C 15  Internal_Error Routine
D 15  Internal_Error Routine
E 15  Internal_Error Routine
F 15  Internal_Error Routine
G 15  Internal_Error Routine
H 15  DSR$Print_TypeCode_Name Routin
I 15  DSR$Print_TypeCode_Name Routin
J 15  DSR$Print_TypeCode_Name Routin
K 15  DSR$Print_TypeCode_Name Routin
L 15  DSR$Print_TypeCode_Name Routin
M 15  DSR$Print_TypeCode_Name Routin
N 15  *** CVRTIM time conversion rou
B 16  *** CVRTIM time conversion rou
C 16  *** CVRTIM time conversion rou
D 16  *** CVRTIM time conversion rou
E 16  *** CVRTIM time conversion rou
F 16  CONVERT BINARY TIME TO ASCII S
G 16  CONVERT BINARY TIME TO ASCII S
H 16  CONVERT ASCII STRING TO BINARY
I 16  CONVERT ASCII STRING TO BINARY
J 16  CONVERT ASCII STRING TO BINARY
K 16  CONVERT ASCII STRING TO BINARY
L 16  CONVERT ASCII STRING TO BINARY
```

```
                                02FC    524          .SBTTL  CONVERT BINARY TIME TO NUMERIC TIME
                                02FC    525 ;+
                                02FC    526 ; EXE$NUMTIM - CONVERT BINARY TIME TO NUMERIC TIME
                                02FC    527 ;
                                02FC    528 ; THIS SERVICE PROVIDES THE CAPABILITY TO CONVERT AN ABSOLUTE OR DELTA TIME
                                02FC    529 ; FROM 64-BIT FORMAT TO INTEGER DATE AND TIME VALUES.
                                02FC    530 ;
                                02FC    531 ; INPUTS:
                                02FC    532 ;
                                02FC    533 ;       NTIMBUF(AP) = ADDRESS OF 7-WORD BUFFER TO RECEIVE CONVERTED DATE AND
                                02FC    534 ;                     TIME VALUES.
                                02FC    535 ;       NTIMADR(AP) = ADDRESS OF 64-BIT TIME VALUE. IF ZERO, THEN THE CURRENT
                                02FC    536 ;                     SYSTEM TIME IS USED. POSITIVE VALUES ARE INTERPRETED AS
                                02FC    537 ;                     ABSOLUTE TIMES AND NEGATIVE VALUES AS DELTA TIMES.
                                02FC    538 ;
                                02FC    539 ; OUTPUTS:
                                02FC    540 ;
                                02FC    541 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CONVERT TO NUMERIC TIME.
                                02FC    542 ;
                                02FC    543 ;               R0 = SS$_ACCVIO - 64-BIT TIME VALUE CANNOT BE READ BY CALLING
                                02FC    544 ;                    ACCESS MODE OR TIME BUFFER CANNOT BE WRITTEN BY
                                02FC    545 ;                    CALLING ACCESS MODE.
                                02FC    546 ;
                                02FC    547 ;               R0 = SS$_IVTIME - SPECIFIED DELTA TIME IS GREATER THAN 9999
                                02FC    548 ;                    DAYS.
                                02FC    549 ;
                                02FC    550 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                02FC    551 ;
                                02FC    552 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                02FC    553 ;-
                                02FC    554
                                02FC    555 EXE$NUMTIM::                                ;CONVERT TO NUMERIC TIME
                      00FC      02FC    556          .WORD   ^M<R2,R3,R4,R5,R6,R7>      ;ENTRY MASK
 57   04 AC   D0               02FE    557          MOVL    NTIMBUF(AP),R7             ;GET ADDRESS OF 7-WORD TIME BUFFER
                                0302    558          IFNOWRT #7*2,(R7),10$              ;CAN TIME BUFFER BE WRITTEN?
    50   01   3C               0308    559          MOVZWL  #SS$_NORMAL,R0             ;ASSUME NORMAL COMPLETION
00000000'EF   7D               030B    560          MOVQ    EXE$GQ_SYSTIME,R1          ;ASSUME TIME NOT SPECIFIED
         51                    0311
 53   08 AC   D0               0312    561          MOVL    NTIMADR(AP),R3             ;GET ADDRESS OF 64-TIME VALUE
         1C   13               0316    562          BEQL    20$                        ;IF EQL NONE SPECIFIED
                               0318    563          IFNORD  #8,(R3),10$                ;CAN 64-BIT TIME VALUE BE READ?
    51   63   7D               031E    564          MOVQ    (R3),R1                    ;GET 64-BIT TIME VALUE
         11   18               0321    565          BGEQ    20$                        ;IF GEQ ABSOLUTE TIME
    52   52   CE               0323    566          MNEGL   R2,R2                      ;NEGATE DELTA TIME VALUE
    51   51   CE               0326    567          MNEGL   R1,R1                      ;
    52   00   D9               0329    568          SBWC    #0,R2                      ;
 04 50   00   E4               032C    569          BBSC    #0,R0,20$                  ;INDICATE DELTA TIME VALUE
    50   0C   3C               0330    570 10$:     MOVZWL  #SS$_ACCVIO,R0             ;SET ACCESS VIOLATION
              04               0333    571          RET                                ;
                               0334    572
                               0334    573 ;
                               0334    574 ; R1 AND R2 CONTAIN 64-BIT ABSOLUTE TIME VALUE IN UNITS OF TENTHS OF MICRO-
                               0334    575 ; SECONDS. CALCULATE DAYS PAST BASE TIME AND FRACTION OF DAY BY DIVIDING
                               0334    576 ; BY 864000000000 WHICH IS THE NUMBER OF TENTHS OF MICROSECONDS IN A DAY.
                               0334    577 ; THE DIVISION IS PERFORMED IN THREE STEPS TO INSURE BOTH QUOTIENT AND
                               0334    578 ; REMAINDER STAY WITHIN 32 BITS.
                               0334    579 ;
```

```
                                0334    580 ; CALCULATE DAYS BY DIVIDING BY 1024 AND THEN 843750000. QUOTIENT IS DAYS
                                0334    581 ; AND REMAINDER IS FRACTION OF DAY.
                                0334    582 ;
                                0334    583
      51    0A    00    EF      0334    584 20$:    EXTZV   #0,#10,R1,R4              ;SAVE REMAINDER FROM NEXT DIVIDE
                        54      0338
        51    F6    8F    79    0339    585         ASHQ    #-10,R1,R1              ;DIVIDE BY 1024
                        51      033D
     324A9A70    8F    7B      033E    586         EDIV    #843750000,R1,R1,R2     ;CALCULATE DAYS AND FRACTION OF DAY
      52    51    51            0344
                                0347    587
                                0347    588 ;
                                0347    589 ; R1 CONTAINS DAYS PAST BASE TIME, R2 PLUS R4 CONTAIN FRACTION OF DAY.
                                0347    590 ; R2 CONTAINS PART OF FRACTION IN UNITS OF 864000000000/1024 AND
                                0347    591 ; R4 CONTAINS REMAINDER IN UNITS OF TENTHS OF MICROSECONDS.
                                0347    592 ;
                                0347    593 ; CALCULATE FRACTION OF DAY IN HUNDREDTHS OF SECONDS BY DIVIDING BY
                                0347    594 ; 100000 WHICH IS THE NUMBER OF TENTHS OF MICROSECONDS IN A HUNDRETH
                                0347    595 ; OF A SECOND.
                                0347    596 ;
                                0347    597
            53    D4            0347    598         CLRL    R3                      ;CLEAR HIGH PART OF DIVIDEND
      52    52    0A    79      0349    599         ASHQ    #10,R2,R2               ;CONVERT BACK TO TENTHS OF MICROSECONDS
            52    54    C8      034D    600         BISL    R4,R2                   ;ADD REMAINDER BACK
       000186A0    8F    7B      0350    601         EDIV    #100000,R2,R5,R2        ;CALCULATE FRACTION OF DAY IN HUNDREDTHS
      52    55    52            0356
                                0359    602
                                0359    603 ;
                                0359    604 ; R1 CONTAINS DAYS PAST THE BASE TIME AND R5 CONTAINS THE FRACTION OF DAY
                                0359    605 ; IN HUNDREDTHS OF A SECOND.
                                0359    606 ;
                                0359    607
     7E 50    00    E3          0359    608         BBCS    #0,R0,70$               ;IF CLR, DELTA TIME SPECIFIED
                                035D    609
                                035D    610 ;
                                035D    611 ; ADD TIME OFFSET SO THAT DAY IS RELATIVE TO 1-JAN-1501.
                                035D    612 ;
                                035D    613
   0001FE98    8F    C0          035D    614         ADDL    #TIMOFF1,R1             ;ADD TIME OFFSET
            51                  0363
                                0364    615
                                0364    616 ;
                                0364    617 ; CALCULATE NUMBER OF QUADRICENTURIES THAT HAVE PAST SINCE 1501.
                                0364    618 ;
                                0364    619
            52    D4            0364    620         CLRL    R2                      ;CLEAR HIGH PART OF DIVIDEND
   00023AB1    8F    7B          0366    621         EDIV    #QUADRIDAYS,R1,R1,R2    ;CALCULATE NUMBER OF QUADRICENTURIES
      52    51    51            036C
                                036F    622
                                036F    623 ;
                                036F    624 ; R1 CONTAINS THE NUMBER OF QUADRICENTURIES AND R2 CONTAINS THE NUMBER OF
                                036F    625 ; DAYS INTO THE NEXT QUADRICENTURY. CALCULATE THE NUMBER OF CENTURIES BY
                                036F    626 ; CONVERTING TO QUARTER DAYS INTO NEXT QUADRICENTURY AND THEN DIVIDING BY
                                036F    627 ; THE AVERAGE NUMBER OF QUARTER DAYS IN A CENTURY.
                                036F    628 ;
                                036F    629
        52    04    C4          036F    630         MULL    #4,R2                   ;CALCULATE NUMBER OF QUARTER DAYS
```

```
                53   D4  0372   631          CLRL    R3                              ;CLEAR HIGH PART OF DIVIDEND
 00023AB1       8F   7B  0374   632          EDIV    #QDAYSPCENT,R2,R2,R3            ;CALCULATE NUMBER OF CENTURIES
       53   52   52       037A
                         037D   633
                         037D   634 ;
                         037D   635 ; R2 CONTAINS THE NUMBER OF CENTURIES AND R3 CONTAINS THE NUMBER OF QUARTER
                         037D   636 ; DAYS INTO THE NEXT CENTURY.
                         037D   637 ;
                         037D   638 ; CALCULATE YEARS BY DISCARDING ANY FRACTION OF A DAY, ADDING 3/4'THS OF A
                         037D   639 ; DAY, AND DIVIDING BY THE AVERAGE NUMBER OF DAYS IN A YEAR. THE LEAP DAY
                         037D   640 ; OF EACH FOUR YEAR CYCLE IS FORCED INTO THE FOURTH YEAR.
                         037D   641 ;
                         037D   642
                54   D4  037D   643          CLRL    R4                              ;CLEAR HIGH PART OF DIVIDEND
           53    03   C8  037F   644          BISL    #3,R3                           ;TRUNCATE FRACTION AND ADD 3/4'THS OF DAY
 000005B5      8F   7B  0382   645          EDIV    #QDAYSPYEAR,R3,R3,R4            ;CALCULATE NUMBER OF YEARS
       54   53   53       0388
                54   04  C6   038B   646          DIVL    #4,R4                           ;CALCULATE NUMBER OF DAYS MINUS ONE
                54   D6  038E   647          INCL    R4                              ;CONVERT TO ACTUAL JULIAN DAY OF YEAR
                         0390   648
                         0390   649 ;
                         0390   650 ; R1 CONTAINS NUMBER OF QUADRICENTURIES.
                         0390   651 ; R2 CONTAINS NUMBER OF CENTURIES.
                         0390   652 ; R3 CONTAINS NUMBER OF YEARS.
                         0390   653 ; R4 CONTAINS JULIAN DAY OF YEAR.
                         0390   654 ;
                         0390   655 ; CALCULATE ACTUAL YEAR.
                         0390   656 ;
                         0390   657
          51    6241     DE   0390   658          MOVAL   (R2)[R1],R1             ;COMBINE CENTURIES AND QUADRICENTURIES
            51    32     C4   0394   659          MULL    #50,R1                  ;CALCULATE NUMBER OF DOUBLE CENTURIES
    51    05DD  C341     3E   0397   660          MOVAW   1501(R3)[R1],R1         ;CALCULATE ACTUAL YEAR
             87    51     B0   039D   661          MOVW    R1,(R7)+                ;STORE YEAR
                         03A0   662
                         03A0   663 ;
                         03A0   664 ; TEST FOR NONLEAP YEAR AND BIAS DAY IF AFTER 28-FEB.
                         03A0   665 ;
                         03A0   666
               51    03   D3   03A0   667          BITL    #3,R1                   ;YEAR MULTIPLE OF 4?
                     14   12   03A3   668          BNEQ    30$                     ;IF NEQ NO
                     52   D4   03A5   669          CLRL    R2                      ;CLEAR HIGH PART OF DIVIDEND
 00000064            8F   7B   03A7   670          EDIV    #100,R1,R1,R2           ;CALCULATE CENTURY AND YEAR IN CENTURY
    52    51         51       03AD
                     52   D5   03B0   671          TSTL    R2                      ;YEAR MULTIPLE OF 100?
                     0C   12   03B2   672          BNEQ    40$                     ;IF NEQ NO
               51    03   D3   03B4   673          BITL    #3,R1                   ;YEAR MULTIPLE OF 400?
                     07   13   03B7   674          BEQL    40$                     ;IF EQL YES
               54    38   D1   03B9   675 30$:     CMPL    #31+28,R4               ;AFTER 28-FEB?
                     02   18   03BC   676          BGEQ    40$                     ;IF GEQ NO
                     54   D6   03BE   677          INCL    R4                      ;ADJUST FOR TABLE BIAS
               51    01   D0   03C0   678 40$:     MOVL    #1,R1                   ;INITIALIZE MONTH
    52    FC37  CF41     9A   03C3   679 50$:     MOVZBL  W^DATETABLE-1[R1],R2    ;GET NUMBER OF DAYS IN MONTH
               54    52   C2   03C9   680          SUBL    R2,R4                   ;SUBTRACT FROM JULIAN DAY
                     04   15   03CC   681          BLEQ    60$                     ;IF LEQ CORRECT MONTH FOUND
          F1   51    0C   F3   03CE   682          AOBLEQ  #12,R1,50$              ;LOOP THROUGH ALL MONTHS
             87    51   B0   03D2   683 60$:     MOVW    R1,(R7)+                ;STORE MONTH
       87    54    52   A1   03D5   684          ADDW3   R2,R4,(R7)+             ;STORE DAY
```

```
                     14   11   03D9   685          BRB     80$                     ;
                                03DB   686
                                03DB   687 ;
                                03DB   688 ; DELTA TIME SPECIFIED - STORE RELATIVE DAY
                                03DB   689 ;
                                03DB   690
                     87   D4   03DB   691 70$:     CLRL    (R7)+                   ;CLEAR YEAR AND MONTH
                  87 51   B0   03DD   692          MOVW    R1,(R7)+                ;STORE DAY
            00002710 8F   D1   03E0   693          CMPL    #10000,R1               ;RELATIVE DAY WITHIN LIMITS?
                     51        03E6
                     06   1A   03E7   694          BGTRU   80$                     ;IF GTRU YES
               50  0184 8F   3C   03E9   695          MOVZWL  #SS$_IVTIME,R0          ;SET INVALID TIME
                          04   03EE   696          RET                             ;
                                03EF   697
                                03EF   698 ;
                                03EF   699 ; R5 CONTAINS FRACTION OF DAY IN HUNDREDTHS OF SECONDS.
                                03EF   700 ;
                                03EF   701 ; CALCULATE HOUR, MINUTE, SECOND, AND HUNDREDTH OF SECOND.
                                03EF   702 ;
                                03EF   703
                  57   08   C0   03EF   704 80$:     ADDL    #8,R7                   ;POINT TWO BYTES PAST END OF BUFFER
                     51   D4   03F2   705          CLRL    R1                      ;CLEAR LOOP INDEX
            52   FC43 CF41   9A   03F4   706 90$:     MOVZBL  W^TIMETABLE[R1],R2      ;GET NEXT UNIT DIVISOR
                     56   D4   03FA   707          CLRL    R6                      ;CLEAR HIGH PART OF DIVIDEND
               55   55   52   7B   03FC   708          EDIV    R2,R5,R5,R6             ;CALCULATE NEXT PART
                     56        0400
                  77   56   B0   0401   709          MOVW    R6,-(R7)                ;STORE NEXT PART
                EC 51   02   F3   0404   710          AOBLEQ  #2,R1,90$               ;LOOP FOR HUNDREDTHS, SECONDS, AND MINUTES
                  77   55   B0   0408   711          MOVW    R5,-(R7)                ;STORE HOUR
                          04   040B   712          RET                             ;
                                040C   713
                                040C   714          .END
```

```
$$T2                    =  00000007     D
ACVTFLG                 =  00000010     D
ATIMADR                 =  0000000C     D
ATIMBUF                 =  00000008     D
ATIMLEN                 =  00000004     D
BLANK                   =  00000020     D
BTIMADR                 =  00000008     D
BTIMBUF                 =  00000004     D
CENTURYDAYS             =  00008EAC     D
COLON                   =  0000003A     D
CONVERT                    000001B3 R   D   02
CVRTIME                    000001E6 R   D   02
DATE                       0000003F R   D   02
DATETABLE                  00000000 R   D   02
DAY                     =  00000C004    D
DELTA                      0000004D R   D   02
EXE$ASCTIM                 00000065 RG  D   02
EXE$BINTIM                 00000106 RG  D   02
EXE$GQ_SYSTIME             ********     X   02
EXE$NUMTIM                 000002FC RG  D   02
HOUR                    =  00000006     D
HUNDREDTH               =  0000000C     D
HYPHEN                  =  0000002D     D
IVTIME                     000001E0 R   D   02
MINUTE                  =  00000008     D
MONTH                   =  00000002     D
MONTHTAB                   0000000C R   D   02
NINE                    =  00000039     D
NTIMADR                 =  00000008     D
NTIMBUF                 =  00000004     D
ONE                     =  00000030     D
PERIOD                  =  0000002E     D
QDAYSPCENT              =  00023AB1     D
QDAYSPYEAR              =  000005B5     D
QUADRIDAYS              =  00023AB1     D
QUADYEARDAYS            =  000005B5     D
SECOND                  =  0000000A     D
SS$_ACCVIO              =  0000000C     D
SS$_IVTIME              =  00000184     D
SS$_NORMAL              =  00000001     D
SYS$FAO                    ********   X   02
SYS$NUMTIM                 ********  GX   02
TIME                       00000052 R   D   02
TIMETABLE                  0000003C R   D   02
TIMOFF1                 =  0001FE98     D
TIMOFF2                 =  00016FEC     D
YEAR                    =  00000000     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	0000040C (1036.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T2	=00000007	286 (1)	273 (1) 286 (1)
ACVTFLG	=00000010	57 (1)	#-253 (1)
ATIMADR	=0000000C	56 (1)	#-239 (1)
ATIMBUF	=00000008	55 (1)	#-234 (1)
ATIMLEN	=00000004	54 (1)	#-287 (1)
BLANK	=00000020	123 (1)	#-324 (1) 368 (1) #-371 (1) 388 (1)
BTIMADR	=00000008	64 (1)	#-521 (1)
BTIMBUF	=00000004	63 (1)	#-320 (1)
CENTURYDAYS	=00008EAC	79 (1)	#-463 (1)
COLON	=0000003A	124 (1)	382 (1) 384 (1)
CONVERT	000001B3-R	395 (1)	#-337 (1) #-367 (1) #-381 (1) #-383 (1)
			#-385 (1) #-387 (1)
CVRTIME	000001E6-R	427 (1)	#-389 (1) #-399 (1)
DATE	0000003F-R	193 (1)	262 (1) #-263 (1)
DATETABLE	00000000-R	148 (1)	#-468 (1) #-679 (1)
DAY	=00000004	136 (1)	#-273 (1) 336 (1) #-428 (1) #-438 (1)
			#-483 (1)
DELTA	00000078-R	194 (1)	#-263 (1) 270 (1) #-271 (1)
EXE$ASCTIM	00000065-R	232 (1)	#-284 (1)
EXE$BINTIM	00000106-R	316 (1)	
EXE$GQ_SYSTIME	00000000-XR		#-560 (1)
EXE$NUMTIM	000002FC-R	555 (1)	
HOUR	=00000006	137 (1)	#-286 (1) 380 (1) #-430 (1) #-494 (1)
HUNDREDTH	=0000000C	140 (1)	#-286 (1) #-366 (1) #-436 (1) #-499 (1)
HYPHEN	=0000002D	125 (1)	#-326.8 (1) 338 (1) #-341 (1) #-353 (1)
IVTIME	000001E0-R	420 (1)	#-342.12 (1) #-351 (1) #-354 (1) #-404 (1)
			#-406 (1) #-408 (1) #-410 (1) #-429 (1)
			#-431 (1) #-433 (1) #-435 (1) #-437 (1)
			#-446 (1) #-449 (1)
MINUTE	=00000008	138 (1)	#-286 (1) #-432 (1) #-495 (1)
MONTH	=00000002	135 (1)	#-260 (1) #-342.14 (1) #-466 (1)
MONTHTAB	0000000C-R	166 (1)	261 (1) #-342.9 (1)
NINE	=00000039	126 (1)	#-405 (1)
NTIMADR	=00000008	71 (1)	#-561 (1)
NTIMBUF	=00000004	70 (1)	#-557 (1)
ONE	=00000030	127 (1)	#-403 (1) #-405 (1)
PERIOD	=0000002E	128 (1)	386 (1)
QDAYSPCENT	=00023AB1	85 (1)	#-632 (1)
QDAYSPYEAR	=000005B5	91 (1)	#-645 (1)
QUADRIDAYS	=00023AB1	97 (1)	#-464 (1) #-621 (1)
QUADYEARDAYS	=000005B5	103 (1)	#-462 (1)
SECOND	=0000000A	139 (1)	#-286 (1) #-434 (1) #-497 (1)
SS$_ACCVIO	=0000000C		#-570 (1)
SS$_IVTIME	=00000184		#-420 (1) #-482 (1) #-695 (1)
SS$_NORMAL	=00000001		#-516 (1) #-559 (1)
SYS$FAO	00000000-XR		273 (1) 286 (1)
SYS$NUMTIM	00000000-XR		246 (1) 330 (1)
TIME	00000052-R	195 (1)	#-271 (1) 283 (1) #-284 (1)
TIMETABLE	0000003C-R	184 (1)	#-706 (1)
TIMOFF1	=0001FE98	110 (1)	#-614 (1)

```
TIMOFF2                         =00016FEC      117    (1)    #-484    (1)
YEAR                            =00000000      134    (1)    #-273    (1)      356    (1)      363    (1)    #-447    (1)
                                                              #-471    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	46 (1)	46 (1)
$FAO_S	2	273 (1)	273 (1) 286 (1)
$NUMTIM_S	1	246 (1)	246 (1) 330 (1)
$PUSHADR	1	246 (1)	246 (1) 273 (1) 286 (1) 330 (1)
$SSDEF	21	46 (1)	46 (1)
IFNORD	1	563 (1)	563 (1)
IFNOWRT	1	558 (1)	558 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.10	00:00:00.49
Command processing	153	00:00:00.80	00:00:02.12
Pass 1	431	00:00:06.68	00:00:09.21
Symbol table sort	0	00:00:00.64	00:00:00.73
Pass 2	186	00:00:02.18	00:00:03.79
Symbol table output	6	00:00:00.06	00:00:00.06
Psect synopsis output	4	00:00:00.02	00:00:00.04
Cross-reference output	33	00:00:00.24	00:00:00.27
Assembler run totals	848	00:00:10.74	00:00:16.73

The working set limit was 1000 pages.
3376 bytes (66 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 446 non-local and 35 local symbols.
738 source lines were read in Pass 1, producing 0 object records in Pass 2.
15 pages of virtual memory were used to define 12 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	2
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	9

500 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CVRTIM/UPDA=(CVRTIM.UPD,CVRTIM.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(2) 56 DECLARATIONS
(3) 83 CVTFILNAM - CONVERT FILE NAME FROM ASCII TO RAD50
(4) 208 STORER50BYTE - CONVERT AND STORE ASCII BYTE TO RAD50
(5) 245 PACKRAD50 - PACK 3 BYTES OF RAD50 CHARACTERS INTO A WORD
(6) 277 ASCIITORAD50 - CONVERT ASCII CHARACTER TO RAD50

```
                        0000     1          .TITLE  CVTFILNAM - Converts ASCII to RAD50
                        0000    .1          .IDENT  '06-02'
-1                      0000     3
                        0000     4  ;
                        0000     5  ;*****************************************************************************
                        0000     6  ;*                                                                           *
                        0000    .1  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983 BY                                  *
-1                      0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
                        0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
                        0000    10  ;*                                                                           *
                        0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                        0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
                        0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
                        0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
                        0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
                        0000    16  ;*  TRANSFERRED.                                                             *
                        0000    17  ;*                                                                           *
                        0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
                        0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
                        0000    20  ;*  CORPURATION.                                                             *
                        0000    21  ;*                                                                           *
                        0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
                        0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000    24  ;*                                                                           *
                        0000    25  ;*                                                                           *
                        0000    26  ;*****************************************************************************
                        0000    27  ;
                        0000    28  ;++
                        0000    29  ; FACILITY:
                        0000    30  ;
                        0000    31  ;       Used by BOOT58 and FILEREAD
                        0000    32  ;
                        0000    33  ; ABSTRACT:
                        0000    34  ;
                        0000    35  ;       Converts an ASCII string file name, type, and version number
                        0000    36  ;       to a 6-word RAD50 file name and type, and a 16-bit binary
                        0000    37  ;       version number.
                        0000    38  ;
                        0000    39  ; ENVIRONMENT:
                        0000    40  ;
                        0000    41  ;       any
                        0000    42  ;
                        0000    43  ; AUTHOR:
                        0000    44  ;
                        0000    45  ;       Carol Peters    22 March 1979
                        0000    46  ;               Extracted from the Release 1 module FILEREAD.MAR; all
                        0000    47  ;               code and definitions are unchanged from Release 1 form.
                        0000    48  ;
                        0000    49  ; MODIFIED BY:
                        0000    50  ;
                        0000    .1  ;       02      Bob Bergazzi    30-Jun-1983     Version 6.12
                        0000    .2  ;               Adapted for VDS
                        0000    .3  ;
                        0000    51  ;       V03-001 TCM0001         Trudy C. Matthews       15-Nov-1982
                        0000    52  ;               Added G^ to call to LIB$CVT_DTB.
                        0000    53  ;
                        0000    54  ;--
```

```
                        0000     56          .SBTTL  DECLARATIONS
                        0000     57
                        0000     58 ;
                        0000     59 ; Include files
                        0000     60 ;
                        0000     61
                        0000     62          $SSDEF                          ; Status code definitions
                        0000     63
                        0000     64 ;
                        0000     65 ; Equated symbols
                        0000     66 ;
                        0000     67
00000061                0000     68          LCA     =       ^O141           ;LOWER CASE A
0000007A                0000     69          LCZ     =       ^O172           ;LOWER CASE Z
00000041                0000     70          UCA     =       ^O101           ;UPPER CASE A
0000005A                0000     71          UCZ     =       ^O132           ;UPPER CASE Z
00000030                0000     72          ZERO    =       ^O60            ;ASCII ZERO
00000039                0000     73          NINE    =       ^O71            ;ASCII NINE
0000002E                0000     74          DOT     =       ^O56            ;ASCII PERIOD
0000003B                0000     75          SEMI    =       ^O73            ;ASCII SEMICOLON
                        0000     76
                        0000     77 ;
                        0000     78 ; PSECT declaration
                        0000     79 ;
                        0000     80
                    00000000     .1          .PSECT  code, exe, nowrt, shr, long     ;                                       [02]
```

```
                                0000     83          .SBTTL  CVTFILNAM - CONVERT FILE NAME FROM ASCII TO RAD50
                                0000     84
                                0000     85  ;++
                                0000     86  ; FUNCTIONAL DESCRIPTION:
                                0000     87  ;
                                0000     88  ;       THIS ROUTINE CONVERTS THE ASCII FILE NAME (NAME, TYPE, VERSION)
                                0000     89  ; TO THE FILE NAME BLOCK FORM OF 3 WORDS RAD50 NAME, 1 WORD OF TYPE
                                0000     90  ; AND 1 WORD OF VERSION.
                                0000     91  ;
                                0000     92  ; CALLING SEQUENCE:
                                0000     93  ;
                                0000     94  ;       CALLG   ARGLIST,FIL$CVTFILNAM
                                0000     95  ;
                                0000     96  ; INPUT PARAMETERS:
                                C000     97  ;
                                0000     98  ;       FILNAM(AP)      =               ;STRING DESCRIPTOR OF FILE NAME STRING
                                0000     99  ;       FILNAMBLK(AP)   =               ;ADDRESS OF 5 WORD BLOCK
                                0000    100  ;                                               3 WORD RAD50 FILE NAME
                                0000    101  ;                                               1 WORD RAD50 FILE TYPE
                                0000    102  ;                                               1 WORD BINARY VERSION
                                0000    103  ;
                                0000    104  ; IMPLICIT INPUTS:
                                0000    105  ;
                                0000    106  ;       NONE
                                0000    107  ;
                                0000    108  ; OUTPUT PARAMETERS:
                                0000    109  ;
                                0000    110  ;       R0 = SYSTEM STATUS CODE
                                0000    111  ;       FILNAME BLOCK FILLED IN
                                0000    112  ;
                                0000    113  ; IMPLICIT OUTPUTS:
                                0000    114  ;
                                0000    115  ;       NONE
                                0000    116  ;
                                0000    117  ; COMPLETION CODES:
                                0000    118  ;
                                0000    119  ;       SS$_NORMAL                      SUCCESSFUL COMPLETION
                                0000    120  ;       SS$_BADFILENAME                 SYNTAX ERROR IN FILE NAME
                                0000    121  ;
                                0000    122  ; SIDE EFFECTS:
                                0000    123  ;
                                0000    124  ;       NONE
                                0000    125  ;
                                0000    126  ; EQUATED SYMBOLS:
                                0000    127  ;
                                0000    128  ;    OFFSETS FROM AP
                                0000    129  ;
00000004                        0000    130          FILNAM          =       4       ;DESCRIPTOR OF ASCII FILE NAME STRING
00000008                        0000    131          FILNAMBLK       =       8       ;ADDRESS OF 5 WORD FILE NAME BLOCK
                                0000    132  ;
                                0000    133  ;    OFFSETS FROM FP
                                0000    134  ;
FFFFFFFD                        0000    135          TYPE            =       -3      ;BEGINNING OF TYPE FIELD
FFFFFFF4                        0000    136          NAME            =       -12     ;BEGINNING OF NAME FIELD
                                0000    137  ;
                                0000    138  ;--
                                0000    139
```

```
                                0000   140 FIL$CVTFILNAM::
                         007C   0000   141         .WORD   ^M<R2,R3,R4,R5,R6>
                      7E  7C    0002   142         CLRQ    -(SP)                   ;RESERVE AND ZERO A
                      7E  D4    0004   143         CLRL    -(SP)                   ;12 BYTE NAME STRING
                55 04 BC  7D    0006   144         MOVQ    @FILNAM(AP),R5          ;R5 = SIZE, R6 = ADR OF STRING
                   55 55  3C    000A    .1         MOVZWL  R5,R5                   ; only low word used                    [02]
                54 F4 AD  9E    000D   145         MOVAB   NAME(FP),R4             ;ADDRESS TO STORE NAME
                   53 09  D0    0011   146         MOVL    #9,R3                   ;UP TO 9 CHARACTERS
                      0F  11    0014   147         BRB     40$
                                0016   148 ;
                                0016   149 ; STORE UP TO 9 CHARACTERS OF RAD50 IN THE BYTE ARRAY ADDRESSED BY R4
                                0016   150 ; IN PREPARATION FOR CONVERTING INTO THE PACKED RAD50 FORMAT
                                0016   151 ;
                                0016   152 20$:
                   50 86  90    0016   153         MOVB    (R6)+,R0                ;GET THE NEXT CHARACTER
                   50 2E  91    0019   154         CMPB    #DOT,R0                 ;IS IT A DOT?
                      0C  13    001C   155         BEQL    FILETYPE                ;BRANCH IF YES
                   50 3B  91    001E   156         CMPB    #SEMI,R0                ;IS IT A SEMICOLON
                      24  13    0021   157         BEQL    VERSION                 ;BRANCH IF YES
                      59  10    0023   158         BSBB    STORER50BYTE            ;CONVERT AND STORE THE CHARACTER
                                0025   159 40$:
                   EE 55  F4    0025   160         SOBGEQ  R5,20$                  ;COUNT THE CHARACTERS IN THE STRING
                      2F  11    0028   161         BRB     BUILDFNB                ;END OF STRING, NO TYPE, NO VERSION
                                002A   162 ;
                                002A   163 ; NOW SET UP THE 3 BYTE ARRAY OF FILE TYPE
                                002A   164 ;
                                002A   165 FILETYPE:
                54 FD AD  9E    002A   166         MOVAB   TYPE(FP),R4             ;ADDRESS OF BYTE ARRAY
                   53 03  D0    C02E   167         MOVL    #3,R3                   ;MAX SIZE OF FILE TYPE
                      0F  11    0031   168         BRB     40$
                                0033   169 20$:
                   50 86  90    0033   170         MOVB    (R6)+,R0                ;GET NEXT CHARACTER
                   50 3B  91    0036   171         CMPB    #SEMI R0                ;IS IT A SEMICOLON?
                      0C  13    0039   172         BEQL    VERSION                 ;BRANCH IF YES
                   50 2E  91    003B   173         CMPB    #DOT,R0                 ;DOT FOR VERSION TOO
                      07  13    003E   174         BEQL    VERSION                 ;BRANCH IF VERSION DELIMITER
                      3C  10    0040   175         BSBB    STORER50BYTE            ;CONVERT AND STORE THE BYTE
                                0042   176 40$:
                   EE 55  F4    0042   177         SOBGEQ  R5,20$                  ;LOOP THROUGH THE CHARACTERS
                      12  11    0045   178         BRB     BUILDFNB                ;END OF STRING, NO VERSION
                                0047   179 ;
                                0047   180 ; VERSION DELIMITER FOUND, CONVERT THE VERSION
                                0047   181 ;
                                0047   182 VERSION:
             08 AC    08  C1    0047   183         ADDL3   #4*2,FILNAMBLK(AP),-(SP) ;ADDRESS TO STORE VERSION
                      7E        004B
                7E    55  7D    004C   184         MOVQ    R5,-(SP)                ;PUSH ADDRESS, PUSH SIZE OF VERSION STRING
                      03  FB    004F   185         CALLS   #3,G^LIB$CVT_DTB        ;CONVERT DECIMAL VERSION STRING TO BINARY
00000000'GF                     0051
                   2A 50  E9    0056   186         BLBC    R0,BADFILNAM            ;BRANCH IF BAD FILE NAME
                                0059   187 ;
                                0059   188 ; NOW PACK THE TEMPORARY BYTE STRING INTO THE RAD50 WORDS
                                0059   189 ;
                                0059   190 BUILDFNB:
             54 08 AC  D0       0059   191         MOVL    FILNAMBLK(AP),R4        ;ADDRESS TO STORE PACKED RAD50
             53 F4 AD  9E       005D   192         MOVAB   NAME(FP),R3             ;ADDRESS OF NAME STRING
                   63  95       0061   193         TSTB    (R3)                    ;ANY NAME GIVEN?
```

```
                06   13   0063   194          BEQL    20$                         ;BRANCH IF NO
                26   10   0065   195          BSBB    PACKRAD50                   ;1ST 3 CHARACTERS
                24   10   0067   196          BSBB    PACKRAD50                   ;2ND 3 CHARACTERS
                22   10   0069   197          BSBB    PACKRAD50                   ;3RD 3 CHARACTERS
                          006B   198 20$:
          08 AC 06   C1   006B   199          ADDL3   #3*2,FILNAMBLK(AP),R4       ;ADDRESS TO STORE FILE TYPE
                54         006F
       53   FD AD   9E   0070   200          MOVAB   TYPE(FP),R3                 ;ADDRESS OF TYPE STRING
                63   95   0074   201          TSTB    (R3)                        ;ANY FILE TYPE PRESENT?
                02   13   0076   202          BEQL    40$                         ;BRANCH IF NOT
                13   10   0078   203          BSBB    PACKRAD50                   ;CONVERT 3 CHARACTERS
                          007A   204 40$:
           50   01   3C   007A   205          MOVZWL  #SS$_NORMAL,R0              ;INDICATE SUCCESS
                     04   007D   206          RET
```

```
                        007E    208          .SBTTL  STORER50BYTE - CONVERT AND STORE ASCII BYTE TO RAD50
                        007E    209 ;++
                        007E    210 ; FUNCTIONAL DESCRIPTION:
                        007E    211 ;
                        007E    212 ;       CONVERT ASCII BYTE TO RAD50 AND STORE IN THE SPECIFIED BYTE ARRAY
                        007E    213 ;
                        007E    214 ; CALLING SEQUENCE:
                        007E    215 ;
                        007E    216 ;       BSBB    STORER50BYTE
                        007E    217 ;
                        007E    218 ; INPUT:
                        007E    219 ;
                        007E    220 ;       R0 = ASCII BYTE
                        007E    221 ;       R3 = SIZE OF BYTE ARRAY TO STORE IN
                        007E    222 ;       R4 = ADDRESS OF BYTE ARRAY TO STORE IN
                        007E    223 ;
                        007E    224 ; OUTPUT:
                        007E    225 ;
                        007E    226 ;       RSB IF SUCCESSFUL, RET WITH ERROR CODE IN R0 IF ERROR
                        007E    227 ;       R3, R4 UPDATED TO REFLECT THE ADDITIONAL BYTE STORED
                        007E    228 ;
                        007E    229 ;--
                        007E    230
                        007E    231          .ENABL  LSB
                        007E    232
                        007E    233 STORER50BYTE:
                  26    10    007E    234          BSBB    ASCIITORAD50            ;CONVERT TO RAD50
                06 53   F4    0080    235          SOBGEQ  R3,20$                  ;BRANCH IF ROOM TO STORE THIS CHARACTER
                        0083    236 BADFILNAM:
           50   0818 8F   3C  0083    237          MOVZWL  #SS$_BADFILENAME,R0     ;RETURN ERROR CODE
                        04    0088    238          RET
                        0089    239 20$:
                84   50   90  0089    240          MOVB    R0,(R4)+                ;STORE IT
                        05    008C    241          RSB                             ;AND RSB SUCCESSFULLY
                        008D    242
                        008D    243          .DSABL  LSB
```

```
                                008D   245          .SBTTL  PACKRAD50 - PACK 3 BYTES OF RAD50 CHARACTERS INTO A WORD
                                008D   246 ;++
                                008D   247 ; FUNCTIONAL DESCRIPTION:
                                008D   248 ;
                                008D   249 ;       PACK 3 BYTES OF RAD50 CHARACTERS INTO A WORD
                                008D   250 ;
                                008D   251 ; CALLING SEQUENCE:
                                008D   252 ;
                                008D   253 ;       BSBB    PACKRAD50
                                008D   254 ;
                                008D   255 ; INPUTS:
                                008D   256 ;
                                008D   257 ;       R3 = RAD50 BYTE STRING ADDRESS
                                008D   258 ;       R4 = ADDRESS OF WORD ARRAY TO STORE IN
                                008D   259 ;
                                008D   260 ; OUTPUTS:
                                008D   261 ;
                                008D   262 ;       R3 UPDATED TO POINT TO NEXT GROUP OF 3 CHARACTERS
                                008D   263 ;       R4 UPDATED TO POINT AT NEXT WORD TO STORE IN
                                008D   264 ;
                                008D   265 ;--
                                008D   266
                                008D   267 PACKRAD50:
          50    83    9A      008D   268          MOVZBL  (R3)+,R0                ;BUILD PACKED WORD IN R0
     50   0640  8F    A4      0090   269          MULW    #40*40,R0               ;START ACCUMULUATING IN R0
          51    83    9A      0095   270          MOVZBL  (R3)+,R1                ;SECOND CHARACTER TO TEMP
          51    28    A4      0098   271          MULW    #40,R1                  ;MULTIPLY BY THE RADIX
          50    51    A0      009B   272          ADDW    R1,R0                   ;AND ACCUMULATE
          51    83    9A      009E   273          MOVZBL  (R3)+,R1                ;LAST CHARACTER
  84      50    51    A1      00A1   274          ADDW3   R1,R0,(R4)+             ;ACCUMULATE AND STORE RESULT
                      05      00A5   275          RSB
```

```
                                00A6   277          .SBTTL  ASCIITORAD50 - CONVERT ASCII CHARACTER TO RAD50
                                00A6   278 ;++
                                00A6   279 ; FUNCTIONAL DESCRIPTION:
                                00A6   280 ;
                                00A6   281 ;       CONVERT ASCII CHARACTER TO ITS RAD50 EQUIVALENT
                                00A6   282 ;
                                00A6   283 ; CALLING SEQUENCE:
                                00A6   284 ;
                                00A6   285 ;       BSBB    ASCIITORAD50
                                00A6   286 ;
                                00A6   287 ; INPUTS:
                                00A6   288 ;
                                00A6   289 ;       R0 = CHARACTER TO CONVERT
                                00A6   290 ;
                                00A6   291 ; OUTPUTS:
                                00A6   292 ;
                                00A6   293 ;       RSB IF SUCCESSFUL, RET WITH ERROR CODE IN R0 IF ERROR
                                00A6   294 ;       R0 = RAD50 EQUIVALENT
                                00A6   295 ;       R1,R2,R3 PRESERVED
                                00A6   296 ;
                                00A6   297 ;--
                                00A6   298
                                00A6   299 ASCIITORAD50:
7A 8F    50    91               00A6   300          CMPB    R0,#LCZ                 ;LOWER CASE ALPHA?
         07    1A               00AA   301          BGTRU   BADFILNAM               ;BRANCH IF BAD RAD50
61 8F    50    91               00AC   302          CMPB    R0,#LCA
         05    19               00B0   303          BLSS    10$                     ;BRANCH IF NOT
50   60 8F    82               00B2   304          SUBB    #LCA-1,R0               ;CONVERT TO ALPHA
               05               00B6   305          RSB                             ;AND RETURN SUCCESSFULLY
                                00B7   306 10$:
5A 8F    50    91               00B7   307          CMPB    R0,#UCZ                 ;UPPER CASE ALPHA?
         C6    14               00BB   308          BGTR    BADFILNAM               ;BRANCH IF BAD RAD50
41 8F    50    91               00BD   309          CMPB    R0,#UCA
         05    19               00C1   310          BLSS    20$                     ;BRANCH IF NOT ALPHA
50   40 8F    82               00C3   311          SUBB    #UCA-1,R0               ;CONVERT TO RAD50 ALPHA
               05               00C7   312          RSB                             ;AND RETURN SUCCESSFULLY
                                00C8   313 20$:
   39    50    91               00C8   314          CMPB    R0,#NINE                ;NUMERIC?
         B6    14               00CB   315          BGTR    BADFILNAM               ;BRANCH IF BAD RAD50
   30    50    91               00CD   316          CMPB    R0,#ZERO
         B1    19               00D0   317          BLSS    BADFILNAM               ;BRANCH IF BAD RAD50
   50    12    82               00D2   318          SUBB    #ZERO-30,R0             ;CONVERT TO RAD50 NUMERIC
               05               00D5   319          RSB                             ;AND RETURN SUCCESSFULLY
                                00D6   320
                                00D6   321          .END
```

```
ASCIITORAD50                    000000A6 R   D  02
BADFILNAM                       00000083 R   D  02
BUILDFNB                        00000059 R   D  02
DOT                           = 0000002E     D
FIL$CVTFILNAM                   00000000 RG  D  02
FILETYPE                        0000002A R   D  02
FILNAM                        = 00000004     D
FILNAMBLK                     = 00000008     D
LCA                           = 00000061     D
LCZ                           = 0000007A     D
LIB$CVT_DTB                     ********   X   02
NAME                          = FFFFFFF4     D
NINE                          = 00000039     D
PACKRAD50                       0000008D R   D  02
SEMI                          = 0000C03B     D
SS$_BADFILENAME               = 00000818     D
SS$_NORMAL                    = 00000001     D
STORER50BYTE                    0000007E R   D  02
TYPE                          = FFFFFFFD     D
UCA                           = 00000041     D
UCZ                           = 0000005A     D
VERSION                         00000047 R   D  02
ZERO                          = 00000030     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	000000D6 (214.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...							
ASCIITORAD50	000000A6-R	299	(6)	#-234	(4)						
BADFILNAM	00000083-R	236	(4)	#-186	(3)	#-301	(6)	#-308	(6)	#-315	(6)
				#-317	(6)						
BUILDFNB	00000059-R	190	(3)	#-161	(3)	#-178	(3)				
DOT	=0000002E	74	(2)	#-154	(3)	#-173	(3)				
FIL$CVTFILNAM	00000000-R	140	(3)								
FILETYPE	0000002A-R	165	(3)	#-155	(3)						
FILNAM	=00000004	130	(3)	#-144	(3)						
FILNAMBLK	=00000008	131	(3)	#-183	(3)	#-191	(3)	#-199	(3)		
LCA	=00000061	68	(2)	#-302	(6)	#-304	(6)				
LCZ	=0000007A	69	(2)	#-300	(6)						
LIB$CVT_DTB	00000000-XR			185	(3)						
NAME	=FFFFFFF4	136	(3)	145	(3)	192	(3)				
NINE	=00000039	73	(2)	#-314	(6)						
PACKRAD50	0000008D-R	267	(5)	#-195	(3)	#-196	(3)	#-197	(3)	#-203	(3)
SEMI	=0000003B	75	(2)	#-156	(3)	#-171	(3)				
SS$_BADFILENAME	=00000818			#-237	(4)						
SS$_NORMAL	=00000001			#-205	(3)						
STORER50BYTE	0000007E-R	233	(4)	#-158	(3)	#-175	(3)				
TYPE	=FFFFFFFD	135	(3)	166	(3)	200	(3)				
UCA	=00000041	70	(2)	#-309	(6)	#-311	(6)				
UCZ	=0000005A	71	(2)	#-307	(6)						
VERSION	00000047-R	182	(3)	#-157	(3)	#-172	(3)	#-174	(3)		
ZERO	=00000030	72	(2)	#-316	(6)	#-318	(6)				

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	62 (2)	62 (2)
$SSDEF	21	62 (2)	62 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.10	00:00:00.29
Command processing	163	00:00:00.74	00:00:01.77
Pass 1	348	00:00:04.68	00:00:06.24
Symbol table sort	0	00:00:00.62	00:00:00.74
Pass 2	73	00:00:01.27	00:00:01.88
Symbol table output	3	00:00:00.05	00:00:00.06
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	20	00:00:00.11	00:00:00.15
Assembler run totals	644	00:00:07.62	00:00:11.18

The working set limit was 1000 pages.
22812 bytes (45 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 423 non-local and 9 local symbols.
325 source lines were read in Pass 1, producing 0 object records in Pass 2.
9 pages of virtual memory were used to define 7 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	4

464 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CVTFILNAM/UPDA=(CVTFILNAM.UPD,CVTFILNAM.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(2) 59 DEASSIGN I/O CHANNEL

```
                        0000     .1          .TITLE  DASSGN  *** DASSGN deassign QIO channel
                        0000     .2          .IDENT  /05-02/
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
                        0000     15 ;*  TRANSFERRED.                                                             *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                        0000     19 ;*  CORPORATION.                                                             *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 26-AUG-76
                        0000     27 ;
                        0000     .1 ;
                        0000     .2 ;               N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
                        0000     .3 ;       01      DIAGNOSTIC SUPERVISOR INTEGRATION
                        0000     .4 ;               Dave Butenhof   13-may-1980, Version 5.4
                        0000     .5 ;       02      Create .UPD file to convert VMS V2.0 source to Supervisor
                        0000     .6 ;               environment
                        0000     .7 ;               Roger Riggs, 19-Jun-1980, Version 5.5
                        0000     .8 ;       03      Put normal in R0 instead of R1
                        0000     28 ;       V03     LMK0002         LEN KAWELL      27-SEP-1979,03-OCT-1979
                        0000     29 ;       Disassociate mailbox if CCB$M_AMB is set in channel status.
                        0000     30 ;
                        0000     31 ;       V02     LMK0001         LEN KAWELL      27-DEC-1978
                        0000     32 ;       Clear DEV$M_RCK, DEV$M_WCK, and DEV$M_SWL when dismounting
                        0000     33 ;       a foreign mounted volume.
                        0000     34 ;
                        0000     .1 ;
                        0000     .2 ;       DS01    M. Baggett      MARCH-8-1983    VDS 6.11
                        0000     .3 ;               Fix truncation error.
                        0000     .4 ;
                        0000     35 ; SYSTEM SERVICE DEASSIGN I/O CHANNEL
                        0000     36 ;
                        0000     37 ; MACRO LIBRARY CALLS
                        0000     38 ;
                        0000     39
                        0000     40          $CCBDEF                         ;DEFINE CCB OFFSETS
                        0000     41          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     42          $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000     43          $IODEF                          ;DEFINE I/O FUNCTION CODES
                        0000     44          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000     45          $PCBDEF                         ;DEFINE PCB OFFSETS
```

```
                        0000     46          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     47          $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000     48          $SSDFF                          ;DEFINE SYSTEM STATUS VALUES
                        0000     49          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000     50
                        0000     51 ;
                        0000     52 ; LOCAL SYMBOLS
                        0000     53 ;
                        0000     54 ; ARGUMENT LIST OFFSET DEFINITIONS
                        0000     55 ;
                        0000     56
                00000004  0000     57 CHAN=4                                  ;I/O CHANNEL NUMBER
                        0000     .1
                00000000  0000     .2          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                        0000     .3          MODNAM  SYSDASSGN
```

```
                                000A     59          .SBTTL  DEASSIGN I/O CHANNEL
                                000A     60 ;+
                                000A     61 ; EXE$DASSGN - DEASSIGN I/O CHANNEL
                                000A     62 ;
                                000A     63 ; THIS SERVICE DEASSIGNS A PREVIOUSLY ASSIGNED I/O CHANNEL AND CLEARS THE
                                000A     64 ; LINKAGE AND CONTROL INFORMATION IN THE CORRESPONDING CHANNEL CONTROL BLOCK.
                                000A     65 ; IF ANY I/O IS OUTSTANDING ON THE CHANNEL IT IS CANCELLED. IF A FILE IS
                                000A     66 ; OPEN ON THE CHANNEL IT IS CLOSED. IF A MAILBOX WAS ASSOCIATED WITH THE
                                000A     67 ; DEVICE WHEN IT WAS ASSIGNED, THE LINKAGE TO THE MAILBUX IS CLEARED. IF THE
                                000A     68 ; THE CHANNEL IS LAST ONE ASSIGNED TO THE DEVICE AND IT IS MARKED FOR DIS-
                                000A     69 ; MOUNT, THEN THE DISMOUNT IS COMPLETED.
                                000A     70 ;
                                000A     71 ; INPUTS:
                                000A     72 ;
                                C00A     73 ;       CHAN(AP) = NUMBER OF THE I/O CHANNEL TO DEASSIGN.
                                000A     74 ;
                                000A     75 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                000A     76 ;
                                000A     77 ; OUTPUTS:
                                000A     78 ;
                                000A     79 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO DEASSIGN CHANNEL.
                                000A     80 ;
                                000A     81 ;               R0 = SS$_IVCHAN - INVALID CHANNEL NUMBER SPECIFIED.
                                000A     82 ;
                                000A     83 ;               R0 = SS$_NOPRIV - SPECIFIED CHANNEL IS NOT ASSIGNED TO A
                                000A     84 ;                    DEVICE OR THE CALLER DOES NOT HAVE SUFFICIENT
                                000A     85 ;                    PRIVILEGE TO ACCESS THE CHANNEL.
                                000A     86 ;
                                000A     87 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                000A     88 ;
                                000A     89 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                000A     90 ;-
                                000A     91
                          00FC  000A     92          .ENTRY  EXE$DASSGN,^M<R2,R3,R4,R5,R6,R7>
00000000'EF   9E  000C     .1          MOVAB   DS$AX_SOFTPCB, R4       ; Software PCB to R4
              54  0012
 55   04 AC   3C  0013     .2          MOVZWL  CHAN(AP),R5             ;GET CHANNEL NUMBER
      50 55   D0  0017     .3          MOVL    R5,R0                   ;COPY I/O CHANNEL NUMBER
00000000'EF   16  001A     .4          JSB     IOC$VERIFYCHAN          ;VERIFY CHANNEL NUMBER                                  [DSO
-3      21 50   E9  0020     96          BLBC    R0,50$                  ;IF LBC INVALID CHANNEL
     56 51      D0  0023     97          MOVL    R1,R6                   ;COPY ADDRESS OF CCB
     57 52      D0  0026     98          MOVL    R2,R7                   ;SAVE CHANNEL INDEX
                    0029     99          $CANCEL_S R5                    ;CANCEL I/O ON CHANNEL
                    0033     .1 20$:
-6            7E   DC  0033    106 30$:     MOVPSL  -(SP)                   ;SAVE CURRENT PROCESSOR STATUS
                    0035    107          SETIPL  #IPL$_ASTDEL            ;RAISE TO AST DELIVERY LEVEL
         0A A6      B5  0038    108          TSTW    CCB$W_IOC(R6)           ;ANY I/O STILL OUTSTANDING?
            08      13  003B    109          BEQL    60$                     ;IF EQL NO
            8F      D5  003D     .1          TSTL    (SP)+                   ; Remove PSL before repeat
-5                  003F    115 40$:     SETIPL  #0                      ;ALLOW INTERRUPTS
            EF      11  0042    116          BRB     20$                     ;
                    04  0044    117 50$:     RET                             ;
                        0045    118 ;
                        0045    119 ; DEASSIGN CHANNEL AND CHECK IF THIS WAS THE LAST CHANNEL ASSIGNED TO DEVICE
                        0045    120 ;
                        0045     .1 60$:
-1          55   66   DO  0045    122          MOVL    CCB$L_UCB(R6),R5        ;GET ASSIGNED DEVICE UCB ADDRESS
```

```
                    09 A6    94   0048   123          CLRB    CCB$B_AMOD(R6)              ;DEASSIGN CHANNEL
                    50 A5    B7   004B   124          DECW    UCB$W_REFC(R5)              ;DECREMENT DEVICE REFERENCE COUNT
                       14    13   004E   125          BEQL    70$                         ;IF EQL LAST REFERENCE
                 50 A5 01    B1   0050   126          CMPW    #1,UCB$W_REFC(R5)           ;UCB REFERENCE COUNT ONE?
                       5E    12   0054   127          BNEQ    120$                        ;IF NEQ NO
              59 34 A5 00'   E1   0056   128          BBC     S^#DEV$V_ALL,UCB$L_DEVCHAR(R5),120$ ;IF CLR, DEVICE NOT ALLOCATED
              28 A5  60 A4   D1   005B   129          CMPL    CCB$L_PID(R4),UCB$L_PID(R5) ;ALLOCATED TO CURRENT PROCESS?
                       52    12   0060   130          BNEQ    120$                        ;IF NEQ NO
                       03    11   0062   131          BRB     80$                         ;
                             0064   132 ;
                             0064   133 ; DEALLOCATE DEVICE AND IF DISMOUNTING, CLEAR MOUNT OPTIONS AND DEALLOCATE VCB
                             0064   134 ;
                    28 A5    D4   0064   135 70$:     CLRL    UCB$L_PID(R5)               ;CLEAR OWNER PROCESS ID
              25 34 A5 00'   E1   0067   136 80$:     BBC     S^#DEV$V_DMT,UCB$L_DEVCHAR(R5),90$ ;IF CLR, NOT MARKED FOR DISMOUNT
              20 34 A5 00'   E1   006C   137          BBC     S^#DEV$V_FOR,UCB$L_DEVCHAR(R5),90$ ;IF CLR, STRUCTURED VOLUME
                       CA    0071   138          BICL    #DEV$M_DMT!DEV$M_FOR!-      ;CLEAR MARKED FOR DISMOUNT, FOREIGN, AND
                             0072   139                  DEV$M_RCK!DEV$M_WCK!-       ;READ/WRITE CHECK, AND
                             0072   140                  DEV$M_SWL!-                 ;SOFTWARE WRITE LOCKED, AND
                00000000'8F        0072   141                  DEV$M_MNT,UCB$L_DEVCHAR(R5) ;MOUNTED
                    34 A5          0077
                 50 30 A5    D0    0079   142          MOVL    UCB$L_VCB(R5),R0            ;GET ADDRESS OF VCB
                       12    13    007D   143          BEQL    90$                         ;IF EQL NONE
                    30 A5    D4    007F   144          CLRL    UCB$L_VCB(R5)               ;CLEAR ADDRESS OF VCB
                00000000'EF  16    0082     1          JSB     EXE$DEANONPAGED             ;DEALLOCATE VCB
-1                  1A A5    B4    0088   146          CLRW    UCB$W_VPROT(R5)             ;CLEAR VOLUME PROTECTION MASK
                  0800 8F    AA    008B   147          BICW    #UCB$M_VALID,UCB$W_STS(R5) ;CLEAR SOFTWARE VOLUME VALID
                    58 A5          008F
                             0091   148 ;
                             0091   149 ; CALL DRIVER'S CANCEL ROUTINE FOR ANY SPECIAL CLEANUP
                             0091   150 ;
                 50  24 A5   D0    0091   151 90$:     MOVL    UCB$L_DDB(R5),R0            ;GET ADDRESS OF DDB
                 50  0C A0   D0    0095   152          MOVL    DDB$L_DDT(R0),R0            ;GET ADDRESS OF DDT
                             0099   153          SETIPL  UCB$B_FIPL(R5)              ;RAISE TO DEVICE FORK IPL
                 53  4C A5   D0    009D   154          MOVL    UCB$L_IRP(R5),R3            ;GET CURRENT I/O PACKET ADDRESS
                     52 57   D0    00A1   155          MOVL    R7,R2                       ;SET CHANNEL INDEX
                 51  0C A0   D0    00A4     1          MOVL    DDT$L_CANCEL(R0),R1         ;GET ADDRESS OF CANCEL I/O ENTRY POINT
              03 51  10      E0    00A8     2          BBS     #16,R1,100$                 ;++++ BRANCH IF SUPERVISOR ABSOLUTE
                 51  50      C0    00AC     3          ADDL    R0,R1                       ;++++ MAKE ABSOLUTE ADDRESS
                     61      16    00AF     4 100$:    JSB     (R1)                        ;CALL CANCEL I/O ROUTINE
                             00B1     5          SETIPL  #IPL$_ASTDEL                ;LOWER TO AST DELIVERY LEVEL
                 50  01      D0    00B4     6 120$:    MOVL    #SS$_NORMAL, R0             ; Set success code
-24          00000000'EF    17    00B7   180          JMP     IOC$UNLOCK                  ;UNLOCK I/O DATA BASE AND RETURN
                             00BD   181
                             00BD   182          .END
```

```
$ER                             = 00000001    D
$MODULE                           00000000 R  D  02
CCB$B_AMOD                        00000009    D
CCB$B_STS                         00000008    D
CCB$C_LENGTH                      00000010    D
CCB$K_LENGTH                      00000010    D
CCB$L_DIRP                        0000000C    D
CCB$L_UCB                         00000000    D
CCB$L_WIND                        00000004    D
CCB$W_IOC                         0000000A    D
CHAN                            = 00000004    D
DDB$B_ACPCLASS                    00000013    D
DDB$B_TYPE                        0000000A    D
DDB$C_LENGTH                      00000034    D
DDB$K_LENGTH                      00000034    D
DDB$L_ACPD                        00000010    D
DDB$L_DDT                         0000000C    D
DDB$L_LINK                        00000000    D
DDB$L_UCB                         00000004    D
DDB$T_DRVNAME                     00000024    D
DDB$T_NAME                        00000014    D
DDB$W_SIZE                        00000008    D
DDT$L_ALTSTART                    0000001C    D
DDT$L_CANCEL                      0000000C    D
DDT$L_FDT                         00000008    D
DDT$L_REGDUMP                     00000010    D
DDT$L_START                       00000000    D
DDT$L_UNITINIT                    00000018    D
DDT$L_UNSOLINT                    00000004    D
DDT$W_DIAGBUF                     00000014    D
DDT$W_ERRORBUF                    00000016    D
DEV$M_DMT                         ********    X   02
DEV$M_FOR                         ********    X   02
DEV$M_MNT                         ********    X   02
DEV$M_RCK                         ********    X   02
DEV$M_SWL                         ********    X   02
DEV$M_WCK                         ********    X   02
DEV$V_ALL                         ********    X   02
DEV$V_DMT                         ********    X   02
DEV$V_FOR                         ********    X   02
DS$AX_SOFTPCB                     ********    X   02
EXE$DASSGN                        0000000A RG D   02
EXE$DEANONPAGED                   ********    X   02
IOC$UNLOCK                        ********    X   02
IOC$VERIFYCHAN                    ********    X   02
IPL$_ASTDEL                     = 00000002    D
PCB$B_ASTACT                      0000000C    D
PCB$B_ASTEN                       0000000D    D
PCB$B_PRI                         0000000B    D
PCB$B_PRIB                        0000002F    D
PCB$B_TYPE                        0000000A    D
PCB$B_WEFC                        0000002E    D
PCB$C_LENGTH                      0000008C    D
PCB$K_LENGTH                      0000008C    D
PCB$L_ARB                         00000084    D
PCB$L_ASTQBL                      00000014    D
PCB$L_ASTQFL                      00000010    D
PCB$L_EFC2P                       00000058    D
PCB$L_EFC3P                       0000005C    D
PCB$L_EFCS                        00000050    D
PCB$L_EFCU                        00000054    D
PCB$L_EFWM                        0000004C    D
PCB$L_JIB                         00000078    D
PCB$L_OWNER                       0000001C    D
PCB$L_PHD                         00000064    D
PCB$L_PHYPCB                      00000018    D
PCB$L_PID                         00000060    D
PCB$L_PQB                         0000004C    D
PCB$L_SQBL                        00000004    D
PCB$L_SQFL                        00000000    D
PCB$L_STS                         00000024    D
PCB$L_UIC                         00000088    D
PCB$L_WSSWP                       00000020    D
PCB$L_WTIME                       00000028    D
PCB$Q_PRIV                        0000007C    D
PCB$T_LNAME                       00000068    D
PCB$T_TERMINAL                    00000044    D
PCB$W_APTCNT                      00000030    D
PCB$W_ASTCNT                      00000038    D
PCB$W_BIOCNT                      0000003A    D
PCB$W_BIOLM                       0000003C    D
PCB$W_DIOCNT                      0000003E    D
PCB$W_DIOLM                       00000040    D
PCB$W_GPGCNT                      00000034    D
PCB$W_GRP                         0000008A    D
PCB$W_MEM                         00000088    D
PCB$W_MTXCNT                      0000000E    D
PCB$W_PPGCNT                      00000036    D
PCB$W_PRCCNT                      00000042    D
PCB$W_SIZE                        00000008    D
PCB$W_STATE                       0000002C    D
PCB$W_TMBU                        00000032    D
PR$_IPL                         = 00000012    D
SIZ...                          = 00000002    D
SS$_NORMAL                      = 00000001    D
SYS$CANCEL                        ********  GX    02
UCB$B_AMOD                        00000053    D
UCB$B_CEX                         00000077    D
UCB$B_CM1                         0000004A    D
UCB$B_CM2                         0000004B    D
UCB$B_DEVCLASS                    00000038    D
UCB$B_DEVTYPE                     00000039    D
UCB$B_DIPL                        00000052    D
UCB$B_DX_SCTCNT                   00000050    D
UCB$B_ERTCNT                      00000070    D
UCB$B_ERTMAX                      00000071    D
UCB$B_FEX                         00000076    D
UCB$B_FIPL                        0000000B    D
UCB$B_LOCSRV                      0000003C    D
UCB$B_OFFNDX                      00000094    D
UCB$B_OFFRTC                      00000095    D
UCB$B_REMSRV                      0000003D    D
UCB$B_SECTORS                     0000003C    D
UCB$B_SLAVE                       00000074    D
```

```
UCB$B_SPR                       00000075     D
UCB$B_STATE                     00000052     D
UCB$B_TRACKS                    0000003D     D
UCB$B_TT_CRFILL                 0000009D     D
UCB$B_TT_DECRF                  000000A1     D
UCB$B_TT_DELFF                  000000A2     D
UCB$B_TT_DESPEE                 000000A0     D
UCB$B_TT_DETYPE                 000000A4     D
UCB$B_TT_LFFILL                 0000009E     D
UCB$B_TT_SPEED                  0000009C     D
UCB$B_TYPE                      0000000A     D
UCB$B_VERTSZ                    0000003F     D
UCB$C_LENGTH                    00000074     D
UCB$C_MB_LENGTH                 00000090     D
UCB$C_TT_LENGTH                 000000BC     D
UCB$K_LENGTH                    00000074     D
UCB$K_MB_LENGTH                 00000090     D
UCB$K_TT_LENGTH                 000000BC     D
UCB$L_AMB                       00000054     D
UCB$L_ASTQBL                    00000010     D
UCB$L_ASTQFL                    0000000C     D
UCB$L_CPID                      0000005C     D
UCB$L_CRB                       00000020     D
UCB$L_DDB                       00000024     D
UCB$L_DEVCHAR                   00000034     D
UCB$L_DEVDEPEND                 0000003C     D
UCB$L_DPC                       00000080     D
UCB$L_DUETIM                    0000005C     D
UCB$L_DX_BFPNT                  0000009C     D
UCB$L_DX_BUF                    00000098     D
UCB$L_DX_RXDB                   000000AC     D
UCB$L_EMB                       00000078     D
UCB$L_FIRST                     00000014     D
UCB$L_FPC                       0000000C     D
UCB$L_FQBL                      00000004     D
UCB$L_FQFL                      00000000     D
UCB$L_FR3                       00000010     D
UCB$L_FR4                       00000014     D
UCB$L_IOQBL                     00000044     D
UCB$L_IOQFL                     00000040     D
UCB$L_IRP                       0000004C     D
UCB$L_LINK                      0000002C     D
UCB$L_LOGADR                    00000064     D
UCB$L_MAXBLOCK                  00000084     D
UCB$L_MB_MBX                    0000007C     D
UCB$L_MB_PORT                   0000008C     D
UCB$L_MB_RAST                   00000078     D
UCB$L_MB_SHB                    00000080     D
UCB$L_MB_WAST                   00000074     D
UCB$L_MB_WIOQBL                 00000088     D
UCB$L_MB_WIOQFL                 00000084     D
UCB$L_MEDIA                     0000008C     D
UCB$L_NT_DATSSB                 00000074     D
UCB$L_NT_INTSSB                 00000078     D
UCB$L_OPCNT                     00000060     D
UCB$L_OWNUIC                    0000001C     D
UCB$L_PID                       00000028     D
UCB$L_RQBL                      00000004     D
UCB$L_RQFL                      00000000     D
UCB$L_SVAPTE                    00000068     D
UCB$L_SVPN                      00000064     D
UCB$L_TT_DECHAR                 000000A8     D
UCB$L_TT_RDUE                   0000008C     D
UCB$L_TT_RTIMOU                 000000B8     D
UCB$L_VCB                       00000030     D
UCB$M_VALID                   = 00000800     D
UCB$T_PARTNER                   0000000C     D
UCB$W_BCNT                      0000006E     D
UCB$W_BCR                       00000096     D
UCB$W_BOFF                      0000006C     D
UCB$W_BUFQUO                    00000018     D
UCB$W_BYTESTOGO                 0000003E     D
UCB$W_CHARGE                    0000004A     D
UCB$W_CYLINDERS                 0000003E     D
UCB$W_DA                        0000008C     D
UCB$W_DC                        0000008E     D
UCB$W_DEVBUFSIZ                 0000003A     D
UCB$W_DEVSTS                    0000005A     D
UCB$W_DIRSEQ                    00000088     D
UCB$W_DSTADDR                   00000018     D
UCB$W_DX_BCR                    000000A4     D
UCB$W_ECT                       00000090     D
UCB$W_EC2                       00000092     D
UCB$W_ERRCNT                    00000072     D
UCB$W_FUNC                      0000007E     D
UCB$W_MB_SEED                   00000000     D
UCB$W_MSGCNT                    00000016     D
UCB$W_MSGMAX                    00000014     D
UCB$W_NT_CHAN                   0000007C     D
UCB$W_OFFSET                    0000008A     D
UCB$W_REFC                      00000050     D
UCB$W_SIZE                      00000008     D
UCB$W_SRCADDR                   0000001A     D
UCB$W_STS                       00000058     D
UCB$W_TT_DESIZE                 000000A5     D
UCB$W_UNIT                      00000048     D
UCB$W_VPROT                     0000001A     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	000000BD (189.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	57.3 (1)	
$MODULE	00000000-R	57.3 (1)	
CCB$B_AMOD	00000009		#-123 (2)
CCB$L_UCB	00000000		#-122 (2)
CCB$W_IOC	0000000A		#-108 (2)
CHAN	=00000004	57 (1)	#-92.2 (2)
DDB$L_DDT	0000000C		#-152 (2)
DDT$L_CANCEL	00000C0C		#-155.1 (2)
DEV$M_DMT	00000000-XR		#-138 (2)
DEV$M_FOR	00000000-XR		#-138 (2)
DEV$M_MNT	00000000-XR		#-141 (2)
DEV$M_RCK	00000000-XR		#-139 (2)
DEV$M_SWL	00000000-XR		#-140 (2)
DEV$M_WCK	00000000-XR		#-139 (2)
DEV$V_ALL	00000000-XR		#-128 (2)
DEV$V_DMT	00000000-XR		#-136 (2)
DEV$V_FOR	00000000-XR		#-137 (2)
DS$AX_SOFTPCB	00000000-XR		92.1 (2)
EXE$DASSGN	0000000A-R	92 (2)	
EXE$DEANONPAGED	00000000-XR		144.1 (2)
IOC$UNLOCK	00000000-XR		180 (2)
IOC$VERIFYCHAN	00000000-XR		92.4 (2)
IPL$_ASTDEL	=00000002	#-107 (2)	#-155.5 (2)
PCB$L_PID	00000060	#-129 (2)	
PR$_IPL	=00000012	#-107 (2)	#-115 (2) #-153 (2) #-155.5 (2)
SS$_NORMAL	=00000001	#-155.6 (2)	
SYS$CANCEL	00000000-XR	99 (2)	
UCB$B_FIPL	0000000B	#-153 (2)	
UCB$L_DDB	00000024	#-151 (2)	
UCB$L_DEVCHAR	00000034	128 (2)	136 (2) 137 (2) #-141 (2)
UCB$L_IRP	0000004C	#-154 (2)	
UCB$L_PID	00000028	#-129 (2)	#-135 (2)
UCB$L_VCB	00000030	#-142 (2)	#-144 (2)
UCB$M_VALID	=00000800	#-147 (2)	
UCB$W_REFC	00000050	#-124 (2)	#-126 (2)
UCB$W_STS	00000058	#-147 (2)	
UCB$W_VPROT	0000001A	#-146 (2)	

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANCEL_S	1	99 (2)	99 (2)
$CCBDEF	1	40 (1)	40 (1)
$DDBDEF	1	41 (1)	41 (1)
$DDTDEF	1	42 (1)	42 (1)
$DEFINI	1	40 (1)	40 (1) 41 (1) 42 (1) 43 (1) 44 (1)
			45 (1) 46 (1) 47 (1) 48 (1) 49 (1)
$IODEF	17	43 (1)	43 (1)
$IPLDEF	1	44 (1)	44 (1)
$PCBDEF	4	45 (1)	45 (1)
$PRDEF	4	46 (1)	46 (1)
$RSNDEF	1	47 (1)	47 (1)
$SSDEF	21	48 (1)	48 (1)
$UCBDEF	10	49 (1)	49 (1)
MODNAM	1	57.3 (1)	57.3 (1)
SETIPL	1	107 (2)	107 (2) 115 (2) 153 (2) 155.5 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.09	00:00:00.24
Command processing	118	00:00:00.71	00:00:01.56
Pass 1	501	00:00:14.09	00:00:19.82
Symbol table sort	0	00:00:01.83	00:00:02.19
Pass 2	85	00:00:02.39	00:00:03.34
Symbol table output	27	00:00:00.17	00:00:00.19
Psect synopsis output	7	00:00:00.03	00:00:00.02
Cross-reference output	18	00:00:00.21	00:00:00.23
Assembler run totals	793	00:00:19.53	00:00:27.62

The working set limit was 1000 pages.
66850 bytes (131 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1174 non-local and 10 local symbols.
171 source lines were read in Pass 1, producing 0 object records in Pass 2.
63 pages of virtual memory were used to define 22 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	6
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	2
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	18

1478 GETS were required to define 18 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DASSGN/UPDA=(DASSGN.UPD,DASSGN.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(2) 42 DECLARATIONS
(3) 126 DD_SELECT - Select correct CPUs for this driver
(4) 164 Console TU58 Driver
(4) 364.3 Character transmission

```
                                .TITLE  CSDDBTDRV - CONSOLE TU58 BOOT DRIVER
                                .IDENT  '06-11'
-2                              
                                ;
                                ;*******************************************************************************
                                ;*                                                                             *
                                ;*  Copyright (c) 1979, 1982, 1983                                             *
-1                              ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                           *
                                ;*                                                                             *
                                ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED     *
                                ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE     *
                                ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER      *
                                ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
                                ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY     *
                                ;*  TRANSFERRED.                                                               *
                                ;*                                                                             *
                                ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE     *
                                ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
                                ;*  CORPORATION.                                                               *
                                ;*                                                                             *
                                ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS     *
                                ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
                                ;*                                                                             *
                                ;*******************************************************************************
                                ;
                                
                                ;++
                                ; FACILITY:     BOOTS
                                ;
                                ; ABSTRACT:
                                ;       This module contains the bootstrap device driver for the
                                ;       console TU58.
                                ;
                                ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
                                ;
                                ; AUTHOR:  Steve Beckhardt,     Creation Date:  1-Nov-1979
                                ;
                                ; MODIFIED BY:
                                ;       11      Bob Bergazzi    May 16, 1983    Version 6.11
                                ;               Changed the order of the .LIB statements.
                                ;
                                ;       10      Marion Baggett, 15-Jul-1982, Version 6.9
                                ;               Increased the timeout value so loading of QALOAD for QA-ing
                                ;               can be done on ZAPPA comet.
                                ;
                                ;       09      - Dave Butenhof, 23-Mar-1982, version 6.7
                                ;               Now that TU58 hardware is fixed (Nebula), remove some
                                ;               hack code
                                ;
                                ;       08      - Dave Butenhof, 10-Feb-1982, version 6.6
                                ;               Fix driver to work correctly on Nebula.  This is mainly
                                ;               a matter of allowing some leeway in READY/DONE bit polling
                                ;               loops to allow console to do something.  Also allow ^C
                                ;               termination of long TU-58 reads.
                                ;
                                ;       V02-007 KTA0037         Kerbey T. Altmann       26-Oct-1981
                                ;               Fix start and size to encompass the entire driver.
```

```
0000     .20 ;
0000     .21 ;       02-006  STJ0125         Steven T. Jeffreys      30-Jul-1981
0000     .22 ;               Added support of MRSP protocol.  For an explantion of the
0000     .23 ;               TU58's operation, including kSP and MRSP protocol, please
0000     .24 ;               see the "TU58 Engineering Specification", document number
0000     .25 ;               TU58-0-0.
0000     .26 ;
0000     .27 ;       02-005  TCM0002         Trudy C. Matthews       29-Jul-1981
0000     .28 ;               Changed all "7ZZ"s to "730"s.
0000     .29 ;
0000     .30 ;       02-004  TCM0001         Trudy C. Matthews       12-Jun-1981
0000     .31 ;               Altered XMIT_TWO_CHARS routine to work on 11/730 processor,
0000     .32 ;               which requires that the high 3 bytes of the value moved to
0000     .33 ;               PR$_CSTD be zero.  Corrected bug in DO_MAPPING routine.
C000     .34 ;               Added CONSDD_START label to mark start of driver.
0000     .35 ;
0000     .36 ;       02-003  SRB0001         Steve Beckhardt         10-Jul-1980
0000     .37 ;               Fixed bugs in change CAS0001
0000     .38 ;
0000     .39 ;       02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000     .40 ;               Change interface to BOOTDRIVR for UBA pruge of buffered
0000     .41 ;               datapath
0000      39 ;
0000      40 ;--
```

```
                        0000     42          .SBTTL  DECLARATIONS
                        0000     43 ;
                        0000     44 ; INCLUDE FILES:
                        0000     45 ;
                        0000      1          .library        /sys$library:lib/       ;                               [11]
                        0000      2          .library        /$ds/           ;                               [11]
                        0000      3          .library        /$diag/         ;                               [11]
                        0000      4
                        0000      5          DsFDef                          ; Define DS$V_ flags            [08]
-1                      0000     47          $BTDDEF                         ; Boot device types
                        0000     48          $IODEF                          ; I/O function codes
                        0000     49          $PRDEF                          ; Processor registers
                        0000     50          $PTEDEF                         ; PTE definitions
                        0000     51          $RPBDEF                         ; RPB offsets
                        0000     52          $SSDEF                          ; Status codes
                        0000     53          $VADEF                          ; Virtual address fields
                        0000     54
                        0000     55 ;
                        0000     56 ; MACROS:
                        0000     57 ;
                        0000     58
                        0000     59 ;
                        0000     60 ; EQUATED SYMBOLS:
                        0000     61 ;
                        0000     62
                        0000     63 ;
                        0000     64 ; CONSOLE TU58 REGISTER DEFINITIONS
                        0000     65 ;
                        0000     66
                        0000     67          $DEFINI DD                      ;START OF REGISTER DEFINITIONS
                        0000     68
                        0000     69          _VIELD  CSRS,0,<-               ;START OF PROC. REG. CSRS DEFS.
                        0000     70                  <,6>,-                  ; UNUSED
                        0000     71                  <IE,,M>,-               ; INTERRUPT ENABLE
                        0000     72                  <DONE,,M>,-             ; DONE
                        0000     73          >
                        0000     74
                        0000     75          _VIELD  CSRD,0,<-               ;START OF PROC. REG. CSRD DEFS.
                        0000     76                  <DATA,8>,-              ; DATA
                        0000     77                  <,7>,-                  ; UNUSED
                        0000     78                  <ERROR,,M>,-            ; ERROR
                        0000     79          >
                        0000     80
                        0000     81          _VIELD  CSTS,0,<-               ;START OF PROC. REG. CSTS DEFS.
                        0000     82                  <BREAK,,M>,-            ; BREAK
                        0000     83                  <,5>,-                  ; UNUSED
                        0000     84                  <IE,,M>,-               ; INTERRUPT ENABLE
                        0000     85                  <READY,,M>,-            ; READY
                        0000     86          >
                        0000     87
                        0000     88          _VIELD  CSTD,0,<-               ;START OF PROC. REG. CSTD DEFS.
                        0000     89                  <DATA,8>,-              ; DATA
                        0000     90          >
                        0000     91
                        0000     92          ;
                        0000     93          ; PROTOCOL FLAGS
                        0000     94          ;
```

```
                    0000     95
00000001            0000     96 DD_DATA = 1
00000002            0000     97 DD_CNTRL = 2
00000004            0000     98 DD_INIT = 4
00000010            0000     99 DD_CONTINUE = 16
                    0000    100
                    0000    101          ;
                    0000    102          ; FUNCTION CODES
                    0000    103          ;
                    0000    104
00000001            0000     .1 DD_Init_Cmd = 1
00000002            0000    105 DD_READ = 2
00000003            0000    106 DD_WRITE = 3
00000040            0000    107 DD_ENDPKT = 64
                    C000    108
                    0000     .1          ;
                    0000     .2          ; SWITCH BITS
                    0000     .3          ;
                    0000     .4
00000003            0000     .5 MRSP_SWITCH = 3
                    0000     .6
                    0000    109          $DEFEND DD
                    0000    110
                    0000    111 ;
-4                  0000    116 ; Boot driver table entry
                    0000    117 ;
                    0000    118
                    0000    119          $BOOT_DRIVER    DEVTYPE = BTD$K_CONSOLE,-  ; Device type (console)
                    0000    120                          ACTION = DD_SELECT,-       ; Action routine
                    0000    121                          SIZE = CONSDD_DRVSIZ,-     ; Driver size
                    0000     .1                          ADDR = CONSDD_START,-     ; Driver start
                    0000     .2                          ENTRY = CONSDD_DRIVER,-   ; Driver entry
                    0000     .3                          DRIVRNAME = DDNAME        ; Driver file name
                    0000     .4
```

```
                                000C   126          .SBTTL  DD_SELECT - Select correct CPUs for this driver
                                0000   127
                                0000   128 ;++
                                0000   129 ; FUNCTIONAL DESCRIPTION:
                                0000   130 ;
                                0000   131 ;       This routine is an action routine called by the boot driver
                                0000   132 ;       select code to determine if this is a cpu that has a TU58
                                0000   133 ;       for a console.
                                0000   134 ;
                                0000   135 ; CALLING SEQUENCE:
                                0000   136 ;
                                0000   137 ;       JSB     DD_SELECT
                                0000   138 ;
                                0000   139 ; INPUT PARAMETERS:
                                0000   140 ;
                                0000   141 ;       EXE$GB_CPUTYPE          Global variable containing cpu type
                                0000   142 ;
                                0000   143 ; OUTPUT PARAMETERS:
                                0000   144 ;
                                0000   145 ;       R0      0 = Do not use this driver
                                0000   146 ;               1 = Use this driver
                                0000   147 ;
                                0000    .1 ;       MRSP    1 if this CPU has a TU58 that speaks MRSP, 0 if not.
                                0000    .2 ;
                                0000    .3 ;               It is assumed that certain CPUs must have MRSP TU58s.
                                0000    .4 ;               Currently, only the 11/730 must have an MRSP TU58.
                                0000    .5 ;               The 11/780 and 11/730 may have MRSP TU58s, but it is
                                0000    .6 ;               not essential that MRSP be used.
                                0000    .7 ;
                                0000    .8 ;--
                                0000    .9
                                0000   .10 CONSDD_START:                          ; Label to mark start of driver
                                0000   .11
                                0000   .12 DD_SELECT:
                00000025'GF   94 0000   .13          CLRB    G^MRSP              ; Assume TU58 does not speak MRSP
-3                 50   D4 0006   151          CLRL    R0                  ; Initialize R0
                                0008   152          CPUDISP <CPU_780,-          ; 11/780
                                0008   153                   CPU_750,-          ; 11/750
                                0008    .1                   CPU_730,-          ; 11/730
-1                              0008   155                   >
                                0016   156 CPU_780:
                                0016   157          RSB                         ; Do not use this driver
                                0017   158
                                0017    .1 CPU_730:
                00000025'GF   D6 0017    .2          INCL    G^MRSP              ; Use MRSP (and fall through to common code)
                                001D    .3
                                001D    .4 CPU_750:
-2                 50   D6 001D   161          INCL    R0                  ; Use this driver
                                001F   162          RSB
```

```
                                0020   164          .SBTTL  Console TU58 Driver
                                0020   165
                                0020   166 ;++
                                0020   167 ;
                                0020   168 ; Inputs:
                                0020   169 ;
                                0020   170 ;       R1      Address of page table for virtual -> physical mapping
                                0020   171 ;       R2      Base VPN of transfer (Bits 29:9 of R10)
                                0020   172 ;       R5      LBN for current piece of transfer
                                0020   173 ;       R8      Size of transfer in bytes
                                0020   174 ;       R9      Address of the RPB
                                0020   175 ;       R10     Starting address of transfer
                                0020   176 ;
                                0020   177 ;       FUNC(AP)  I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                                C020   178 ;       MODE(AP)  Address interpretation mode:
                                0020   179 ;                    0 -> Physical,  1 -> Virtual
                                0020   180 ;
                                0020   181 ; Outputs:
                                0020   182 ;
                                0020   183 ;       R0      Status code:
                                0020   184 ;
                                0020   185 ;               SS$_NORMAL      Successful transfer
                                0020   186 ;               SS$_BUFBYTALI   Odd byte count
                                0020    .1 ;               SS$_CTRLERR     Fatal controller error
                                0020   187 ;--
                      00000010  0020   188 FUNC = 16
                      00000014  0020   189 MODE = 20
                                0020   190
                                0020   191
                                0020   192 ;
                                0020    .1 ; OWN STORAGE:
-1                              0020   194 ;
                                0020   195
                      00000000  0020   196 STKPTR: .LONG   0                       ; Saved stack pointer
                            00  0024   197 INIT:   .BYTE   0                       ; TU58 initialized flag
                            00  0025    .1 MRSP:   .BYTE   0                       ; MRSP protocol flag
                  49 52 44 44 44 00'    0026   198 DDNAME: .ASCIC  /DDDRIVER.EXE/          ; Driver file name
                  58 45 2E 52 45 56     002C
                                45      0032
                                0C      0026
                                0033   199
                                0033   200 CONSDD_DRIVER:
                      0D06 8F   BB      0033   201         PUSHR   #^M<R1,R2,R8,R10,R11>   ; Save input registers
                        0A 58   E9      0037    .1         BLBC    R8,1$                   ; Branch if even byte count
                50    030C 8F   3C      003A    .2         MOVZWL  #SS$_BUFBYTALI,R0       ; Error - odd byte count
                      0D06 8F   BA      003F    .3         POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers
                                05      0043    .4         RSB
                  D8 AF   5E    D0      0044    .5 1$:     MOVL    SP,STKPTR               ; Save stack pointer
-4                              0048   206
                                0048   207 ;
                                0048   208 ; There are 4 possibilities concerning mapping:  the I/O can be
                                0048   209 ; done virtual or physical (MODE(AP)) and we can be executing virtual
                                0048   210 ; or physical (contents of processor register PR$_MAPEN).  If both
                                0048   211 ; modes match, then we can just copy data to/from the user buffer.
                                0048   212 ; If the I/O is to be done virtual and we are executing physical
                                0048   213 ; then the buffer address has to be translated using the page table
                                0048   214 ; pointed to by R1.  If the I/O is to be done physical and we are
```

```
                                0048   215 ; executing virtual then we have to double map the buffer using a spare
                                0048   216 ; PTE.  At this point, we just compute a mapping switch in R11 as follows:
                                0048   217 ;
                                0048   218 ;       0       Both modes match, just copy the data
                                0048   219 ;       1       Do virtual -> physical translation using page table
                                0048   220 ;       -1      Do physical -> virtual mapping using a spare PTE
                                0048   221 ;
                5B    38    DB  0048   222         MFPR    #PR$_MAPEN,R11          ; Get mapping enabled switch
                5B    5B    CE  004B   223         MNEGL   R11,R11                 ; Negate it
             5B   14    AC    C0  004E   224         ADDL    MODE(AP),R11            ; Add I/O mode switch
                                0052   225
                56    51    7D  0052   226         MOVQ    R1,R6                   ; R6 = Addr. of page tbl, R7 = Base VPN
                   0201    30  0055   227         BSBW    DO_MAPPING              ; Initialize mapping if required
                                0058    .1
                   CA AF    95  0058    .2         TstB    Mrsp                    ; If Nebula, turn off clock
                      07    13  005B    .3         BEql    3$                      ; Branch if not Nebula
                80000080 8F    DA  005D    .4         Mtpr    #<1@31>!<1@7>, #Pr$_Iccs; Turn off clock
                      18        0063
                   BD AF    95  0064    .5 3$:     TSTB    INIT                    ; Is it necessary to initialize TU58?
                      03    13  0067    .6         BEql    5$                      ; Yes
                    0021    31  0069    .7         BrW     10$                     ; No
                                006C    .8 5$:
-3                              006C   231         ;
                                006C   232         ; Initialize TU58.  This code is executed the first time the
                                006C   233         ; driver is called and the next time the driver is called
                                006C   234         ; after exiting with an error.
                                006C   235         ;
                                006C   236
                1F    01    DA  006C   237         MTPR    #CSTS_M_BREAK,#PR$_CSTS ; Set BREAK bit in trans. csr
                      52    D4  006F   238         CLRL    R2                      ; R2 contains null characters
                   015D    30  0071   239         BSBW    XMIT_TWO_CHARS          ; Send two nulls
                1F    00    DA  0074   240         MTPR    #0,#PR$_CSTS            ; Clear BREAK bit
                50    1D    DB  0077   241         MFPR    #PR$_CSRD,R0            ; Clear receive buffer
             52 0404  8F    3C  007A   242         MOVZWL  #DD_INIT@8+DD_INIT,R2   ; R2 contains two INIT characters
                   014F    30  007F   243         BSBW    XMIT_TWO_CHARS          ; Send two INIT characters
                   0198    30  0082   244         BSBW    RECV_ONE_CHAR           ; Receive a character
                10    52    91  0085   245         CMPB    R2,#DD_CONTINUE         ; Is it continue?
                      4D    12  0088    .1         BNeq    32$                     ; No, error
-1                 97 AF    96  008A   247         INCB    INIT                    ; So that we don't execute this code again
                                008D   248
                                008D   249 ;
                                008D   250 ; Perform the I/O transfer.  Register usage is:
                                008D   251 ;
                                008D   252 ;       R0              Scratch
                                008D   253 ;       R1              Loop counter
                                008D   254 ;       R2              Character to/from TU58
                                008D   255 ;       R3              Character received from TU58
                                008D   256 ;       R4              Checksum
                                008D   257 ;       R5              Logical block number
                                008D   258 ;       R6              Address of page table
                                008D   259 ;       R7              Virtual page number of buffer
                                008D   260 ;       R8              Size of remaining buffer (in bytes)
                                008D   261 ;       R9              Address of RPB
                                008D   262 ;       R10             Address of current spot in buffer
                                008D   263 ;       R11             Mapping switch
                                008D   264 ;
                                008D    .1 ;
```

```
                                008D   265          ASSUME  DD_WRITE  EQ  DD_READ+1
                                008D   266          ASSUME  DD_DATA  EQ  1
                                008D   267
                          54    D4   008D   268 10$:    CLRL    R4                      ; Clear checksum
                    52  0A02  8F   3C   008F   269          MOVZWL  #10a8+DD_CNTRL,R2        ; Command packet and byte count
                        0134  30   0094   270          BSBW    XMIT_TWO_UPDSUM         ; Send them
                        52    02   9A   0097   271          MOVZBL  #DD_READ,R2             ; Assume read command
                    21    10  AC   D1   009A   272          CMPL    FUNC(AP),#IO$_READLBLK  ; Is it a read?
                              02   13   009E   273          BEQL    20$                     ; Yes
                              52   D6   00A0   274          INCL    R2                      ; No, convert to write command
                        0126  30   00A2   275 20$:    BSBW    XMIT_TWO_UPDSUM         ; Send command and modifier (0)
                                   00A5     .1 ;
                                   00A5     .2 ;       The unit number of the console boot device is always zero.
                                   00A5     .3 ;       This is a result of the 11/750 console implementation, where
                                   00A5     .4 ;       logical device CSA1: is actually physical device DDA0:.
                                   00A5     .5 ;
                          52    D4   00A5     .6          CLRL    R2                      ; Set Unit # and Switch fields
                    FF7A  CF   95   00A7     .7          TSTB    MRSP                    ; Does this TU58 speak MRSP?
                              04   13   00AB     .8          BEQL    25$                     ; Branch if not
                    00  52    0B   E2   00AD     .9          BBSS    #<MRSP_SWITCH+8>,R2,25$ ; Set MRSP switch
                        011?  30   00B1   .10 25$:    BSBW    XMIT_TWO_UPDSUM         ; Send unit # and switches (0)
                              52   D4   00B4   .11          CLRL    R2                      ; Set sequence number (unused field)
-2                      0112  30   00B6   278          BSBW    XMIT_TWO_UPDSUM         ; Send sequence number (0)
                        52    58   3C   00B9   279          MOVZWL  R8,R2                   ; Byte count
                              010C 30   00BC   280          BSBW    XMIT_TWO_UPDSUM         ; Send it
                        52    55   3C   00BF   281          MOVZWL  R5,R2                   ; Block number
                              0106 30   00C2   282          BSBW    XMIT_TWO_UPDSUM         ; Send it
                        52    54   3C   00C5   283          MOVZWL  R4,R2                   ; Checksum
                              0106 30   00C8   284          BSBW    XMIT_TWO_CHARS          ; Send it
                    21    10  AC   D1   00CB   285          CMPL    FUNC(AP),#IO$_READLBLK  ; Is it a read command?
                              42   13   00CF   286          BEQL    50$
                                   00D1   287
                                   00D1   288 ;
                                   00D1   289 ; Do a write to the TU58
                                   00D1   290 ;
                              0149 30   00D1   291 30$:    BSBW    RECV_ONE_CHAR           ; Receive one character
                          10  52   91   00D4   292          CMPB    R2,#DD_CONTINUE         ; Is it CONTINUE?
                              12   12   00D7   293 32$:    BNEQ    52$                     ; No, error
                              54   D4   00D9   294          CLRL    R4                      ; Clear checksum
                    51    80  8F   9A   00DB   295          MOVZBL  #128,R1                 ; Number of bytes to send
                        58    51   D1   00DF   296          CMPL    R1,R8                   ; Is it less than remaining byte count?
                              03   1F   00E2   297          BLSSU   35$                     ; Yes
                        51    58   D0   00E4   298          MOVL    R8,R1                   ; No, use remaining byte count instead
                52      51    08   78   00E7   299 35$:    ASHL    #8,R1,R2                ; Put byte count in second byte
                              52   D6   00EB   300          INCL    R2                      ; Flag byte = DD_DATA
                              00DB 30   00ED   301          BSBW    XMIT_TWO_UPDSUM         ; Send flag byte and byte count
                        51    02   C6   00F0   302          DIVL    #2,R1                   ; Convert byte count to word count
                              0144 30   00F3   303 40$:    BSBW    GETBYTE                 ; Get a byte from memory in R3
                        52    53   D0   00F6   304          MOVL    R3,R2                   ; Save byte in R2
                              013E 30   00F9   305          BSBW    GETBYTE                 ; Get another byte from memory in R3
                08    08    53   F0   00FC   306          INSV    R3,#8,#8,R2             ; Put it in second byte of R2
                              52             0100
                              00C7 30   0101     .1          BSBW    XMIT_TWO_UPDSUM         ; Send two bytes
                          EC  51   F5   0104     .2          SOBGTR  R1,40$                  ; Repeat until byte count is 0
                        52    54   3C   0107     .3          MOVZWL  R4,R2                   ; Get checksum
                              00C4 30   010A     .4          BSBW    XMIT_TWO_CHARS          ; Send it
-4                            58   D5   010D   311          TSTL    R8                      ; Any more data to send?
```

```
                     C0   12  010F   312          BNEQ    30$                     ; Yes, send another packet
                     30   11  0111   313          BRB     END_OF_DATA             ; No, get End packet
                              0113   314
                              0113   315 ;
                              0113   316 ; Do a read from TU58
                              0113   317 ;
                     54   D4  0113   318 50$:     CLRL    R4                      ; Clear checksum
                   00F2   30  0115    .1          BSBW    RECV_TWO_UPDSUM         ; Get flag byte in R3, byte count in R2
                01   53   91  0118    .2          CMPB    R3,#DD_DATA             ; Is this a data packet?
                     2E   12  011B    .3 52$:     BNEQ    DD_ERROR2               ; No, error
              51 52   02   C7  011D    .4          DIVL3   #2,R2,R1                ; Convert byte count to word count in R1
                   00E6   30  0121    .5 55$:     BSBW    RECV_TWO_UPDSUM         ; Receive next two data bytes in R3, R2
-5                 011E   30  0124   324          BSBW    PUTBYTE                 ; Store first byte in memory
                53   52   9A  0127   325          MOVZBL  R2,R3                   ; Move second byte to R3
                   0118   30  012A   326          BSBW    PUTBYTE                 ; Store second byte in memory
                 F1 51    F5  012D   327          SOBGTR  R1,55$                  ; Repeat until byte count is 0
                   00E5   30  0130    .1          BSBW    RECV_TWO_CHARS          ; Get checksum in R3, R2
-1            08 08   52   F0  0133   329          INSV    R2,#8,#8,R3             ; Assemble checksum into one word
                     53       0137
                54   53   B1  0138   330          CMPW    R3,R4                   ; Is checksum correct?
                     79   12  013B   331          BNEQ    DD_ERROR                ; No, error
                     3B   10  013D    .1          BsbB    DD_CheckControlC        ; Check for control C                   [08]
                     58   D5  013F   332          TSTL    R8                      ; Anymore data to receive?
                     D0   12  0141   333          BNEQ    50$                     ; Yes, receive next data packet
                              0143   334
                              0143   335 ;
                              0143   336 ; We've sent or received all the data.  Make sure we receive an end
                              0143   337 ; packet with a success code.
                              0143   338 ;
                              0143   339 END_OF_DATA:
                     54   D4  0143   340          CLRL    R4                      ; Clear checksum
                   00C2   30  0145    .1          BSBW    RECV_TWO_UPDSUM         ; Get flag byte and byte count
                02   53   91  0148    .2          CMPB    R3,#DD_CNTRL            ; Is it a command packet?
                              014B    .3 DD_ERROR2:                               ; (extension for branch to error)
                     69   12  014B    .4          BNEQ    DD_ERROR                ; No, error
                   00BA   30  014D    .5          BSBW    RECV_TWO_UPDSUM         ; Get opcode and success/failure byte
-4              40 8F   53   91  0150   345          CMPB    R3,#DD_ENDPKT           ; Is it an end packet?
                     60   12  0154   346          BNEQ    DD_ERROR                ; No, error
                     52   D5  0156   347          TSTL    R2                      ; Is it success?
                     5C   19  0158   348          BLSS    DD_ERROR                ; No, error
                51   04   D0  015A   349          MOVL    #4,R1                   ; Read remainder of packet
                   00AA   30  015D    .1 10$:     BSBW    RECV_TWO_UPDSUM         ;
                 FA 51    F5  0160    .2          SOBGTR  R1,10$                  ;
                   00B2   30  0163    .3          BSBW    RECV_TWO_CHARS          ; Read checksum in R3, R2
-3            08 08   52   F0  0166   353          INSV    R2,#8,#8,R3             ; Form checksum in R3
                     53       016A
                54   53   B1  016B   354          CMPW    R3,R4                   ; Is checksum correct?
                     46   12  016E   355          BNEQ    DD_ERROR                ; No, error
                     2C   10  0170    .1          BsbB    Reset_Clock             ;
                50   01   3C  0172    .2          MOVZWL  #SS$_NORMAL,R0          ; Return success                        [08]
                0D06 8F   BA  0175    .3          POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers
                     05   0179    .4          RSB
                              017A    .5
                              017A    .6 DD_CheckControlC:                        ; Check for control C                   [08]
          00000000'EF   16  017A    .7          Jsb     L^Kb_Poll               ; Poll keyboard                         [08]
                     00   E1  0180    .8          BbC     S^#DS$V_CtrlC, -        ; If ^C bit                             [08]
          00000000'EF       0182
```

```
                    15      0187   .9                       L^Ds$GL_Flags, 20$      ; .. isn't set, continue                  [08]
              FE98  CF  94  0188   .10          ClrB        Init                    ; Make sure to re-init Tu58 later         [08]
                    10  10  018C   .11          BsbB        Reset_Clock             ;                                         [08]
          5E  FE8E  CF  D0  018E   .12          MOVL        STKPTR,SP               ; Restore stack pointer                   [08]
              0D06  8F  BA  0193   .13          POPR        #^M<R1,R2,R8,R10,R11>   ; Restore registers                       [08]
          50  0651  8F  3C  0197   .14          MovZWL      #SS$_ControlC, R0       ; Set status                              [08]
                    04  019C   .15          Ret                                 ; Return from BOO$QIO, to prevent         [08]
                        019D   .16                                                ;  retry of the operation.                [08]
                    05  019D   .17 20$:      Rsb                                 ; No ^C, just return                      [08]
                        019E   .18
                        019E   .19 Reset_Clock:
              FE83  CF  95  019E   .20          TstB        Mrsp
                    0A  13  01A2   .21          BEql        10$
                    3F  BB  01A4   .22          PushR       #^M<R0,R1,R2,R3,R4,R5>
     00000000'EF      16  01A6   .23          Jsb         L^Clk$Re_Init
                    3F  BA  01AC   .24          PopR        #^M<R0,R1,R2,R3,R4,R5>
                    05  01AE   .25 10$:      Rsb
                        01AF   .26
                        01AF   .27 .Enable LSB                                    ;                                         [08]
                        01AF   .28
                        01AF   .29 DD_TimeOut:                                    ; Operation timed out                     [08]
          50  022C  8F  3C  01AF   .30          MovZWL      #SS$_TimeOut, R0        ; Set timeout code                        [08]
                    05  11  01B4   .31          BrB         10$                     ; Join common                             [08]
                        01B6   .32
                        01B6   .33 DD_ERROR:
          50  0054  8F  3C  01B6   .34          MOVZWL      #SS$_CTRLERR,R0         ; Set failure status                      [08]
                        01BB   .35
              FE65  CF  94  01BB   .36 10$:      CLRB        INIT                    ; Initialize TU58 on next entry           [08]
                    DD  10  01BF   .37          BsbB        Reset_Clock             ;
-5        5E  FE5B  CF  D0  01C1   361          MOVL        STKPTR,SP               ; Restore stack pointer
              0D06  8F  BA  01C6   362          POPR        #^M<R1,R2,R8,R10,R11>   ; Restore registers
                    05  01CA   363          RSB                                 ; Return and retry
                        01CB   364
                        01CB    .1 .DisAble LSB                                   ;                                         [08]
```

```
                                01CB    .3 .Subtitle       Character transmission
                                01CB   365
                                01CB   366 ;++
                                01CB   367 ; XMIT_TWO_UPDSUM - Transmit two characters and update checksum
                                01CB   368 ; XMIT_TWO_CHARS - Transmit two characters
                                01CB   369 ; XMIT_ONE_CHAR - Transmit one character
                                01CB   370 ;
                                01CB   371 ; Inputs:
                                01CB   372 ;       R2      Contains one or two characters in low bytes
                                01CB   373 ;       R4      Checksum so far
                                01CB   374 ;
                                01CB   375 ; Outputs:
                                01CB   376 ;       R4      Updated checksum
                                01CB   377 ;--
                                01CB   378
                                01CB   379 XMIT_TWO_UPDSUM:
                          54    01CB   380          ADDW    R2,R4                   ; Add two characters to checksum
                          54    01CE   381          ADWC    #0,R4                   ; Add carry
                                01D1   382
                                01D1   383 XMIT_TWO_CHARS:
                          52    01D1    .1          PUSHL   R2                      ; Save input on stack.
                          52    01D3    .2          MOVZBL  R2,R2                   ; Zero high bytes of data.
                          05    01D6    .3          BSBB    XMIT_ONE_CHAR           ; Transmit one character
                       8E F8    01D8    .4          ASHL    #-8,(SP)+,R2            ; Retrieve second character
                          52    01DC
-2                              01DD   386                                          ; and fall through to transmit it
                                01DD   387
                                01DD   388 XMIT_ONE_CHAR:
                       50 1E    01DD    .1          MovZBL  #Pr$_CsTs, R0           ; Number of register to poll        [08]
                          06    01E0    .2          BsbB    Wait_Ready              ; Wait for bit 7 to set             [08]
                       1F 52    01E2    .3          MTPR    R2,#PR$_CSTD            ; Transmit character
                          05    01E5    .4          RSB
                                01E6    .5
                                01E6    .6 DD_Error1:                               ; Extension to DD_Error             [08]
                             CE 01E6    .7          BrB     DD_Error                ; Just chain on to it               [08]
                                01E8    .8
                                01E8    .9 ;++
                                01E8   .10 ; Wait_Ready - wait for done/ready bit in specified IPR to set
                                01E8   .11 ;
                                01E8   .12 ; Inputs:
                                01E8   .13 ;       R0      Number of processor register to poll
                                01E8   .14 ;
                                01E8   .15 ; Outputs:
                                01E8   .16 ;       None
                                01E8   .17 ;
                                01E8   .18 ; If desired bit does not set within the timeout period, a direct
                                01E8   .19 ; exit is made from the driver through DD_TIMEOUT.
                                01E8   .20 ;
                                01E8   .21 ;--
                                01E8   .22
                                01E8   .23 Wait_Ready:                                  ; Wait for bit              [08]
                          06    01E8   .24          PushR   #^M<R1, R2>               ; Save R1 for loop counter  [08]
                    001F0000 8F 01EA   .25          MovL    #^X1F0000, R1             ; Fetch value               [10]
                          51    01F0
'01'01'01'01'01'01'             01F1   .26 10$:     .Byte   1[10]                     ; Nops to free console      [08]
                    01'01'01'01 01F7
                       52 50    01FB   .27          Mfpr    R0, R2                    ; Read console CSR          [08]
```

```
                05 52    07   E0  01FE    .28          BbS     #CsRs_V_Done, R2, 20$          ; Exit loop if ready            [08]
                   EC 51 F5   0202    .29          SobGtr  R1, 10$                        ; Otherwise, loop               [08]
                      A8 11   0205    .30          BrB     DD_TimeOut                     ; Timeout if drop here          [08]
                      06 BA   0207    .31 20$:     PopR    #^M<R1, R2>                    ; Restore register              [08]
                      05      0209    .32          Rsb                                    ; Return to caller              [08]
-6                            020A    395 ;++
                              020A    396 ; RECV_TWO_UPDSUM - Receive two characters and update checksum
                              020A    397 ; RECV_TWO_CHARS - Receive two characters
                              020A    398 ; RECV_ONE_CHAR - Receive one character
                              020A    399 ;
                              020A    400 ; Inputs:
                              020A    401 ;       R4      Checksum so far
                              020A    402 ;
                              020A    403 ; Outputs:
                              020A    404 ;       R2      Most recently received character
                              020A    405 ;       R3      Previous character in low byte
                              020A    406 ;--
                              020A    407
                              020A    408 RECV_TWO_UPDSUM:
                      0C 10   020A    409          BSBB    RECV_TWO_CHARS                 ; Receive two characters in R3,  R2
                08    08 52 F0  020C    410          INSV    R2,#8,#8,R3                    ; Put second character into R3
                         53     0210
                      54 53 A0  0211    411          ADDW    R3,R4                          ; Add two characters to checksum
                      54 00 D8  0214    412          ADWC    #0,R4                          ; Add carry to checksum
                         05     0217    413          RSB
                              0218    414
                              0218    415 RECV_TWO_CHARS:
                   03 10        0218    416          BSBB    RECV_ONE_CHAR                  ; Get one character in R2
                53 52 9A        021A    417          MOVZBL  R2,R3                          ; Save first character in R3 and
                              021D    418                                                 ; fall through to get second character
                              021D    419
                              021D    420 RECV_ONE_CHAR:
                50 1C 9A        021D    .1           MovZBL  #Pr$_CsRs, R0                  ; Get number of register        [08]
                   C6 10        0220    .2           BsbB    Wait_Ready                     ; Wait for ready to set         [08]
                52 1D DB        0222    .3           MFPR    #PR$_CSRD,R2                   ; Get next character
                FDFC CF 95      0225    .4           TSTB    MRSP                           ; Does this TU58 speak MRSP?
                      0A 13     0229    .5           BEQL    20$                            ; Branch if not
                50 1E DB        022B    .6 10$:      MFPR    #PR$_CSTS,R0                   ; Get transmit status
             F9 50 07 E1        022E    .7           BBC     #CSTS_V_READY,R0,10$           ; Loop until ready
                1F 10 DA        0232    .8           MTPR    #DD_CONTINUE,#PR$_CSTD         ; Send CONTINUE character
             AD 52 0F E0        0235    .9 20$:      BBS     #CSRD_V_ERROR,R2,DD_ERROR1     ; Branch if data overrun
-4                            0239    425          RSB
                              023A    426
                              023A    427
                              023A    428 ;++
                              023A    429 ; GETBYTE - Subroutine to get a byte from memory
                              023A    430 ; PUTBYTE - Subroutine to store a byte in memory
                              023A    431 ;
                              023A    432 ;       These two subroutines do two things special:
                              023A    433 ;
                              023A    434 ;               1)  Since the floppy always reads or writes 128 bytes
                              023A    435 ;                   these routines simply return if the byte count is zero.
                              023A    436 ;               2)  These routines take care of page boundaries if
                              023A    437 ;                   mapping is required.
                              023A    438 ;
                              023A    439 ; Inputs:
                              023A    440 ;       R3      Byte to store (PUTBYTE)
```

```
                        023A   441 ;       R6      Address of page table
                        023A   442 ;       R7      Virtual page number of buffer
                        023A   443 ;       R8      Size of remaining buffer (in bytes)
                        023A   444 ;       R10     Address of current spot in buffer
                        023A   445 ;       R11     Mapping switch:
                        023A   446 ;                       -1   Do physical -> virtual map
                        023A   447 ;                       0    No mapping required
                        023A   448 ;                       1    Do virtual -> physical translation
                        023A   449 ;
                        023A   450 ; Outputs:
                        023A   451 ;       R3      Byte fetched from memory (GETBYTE)
                        023A   452 ;--
                        023A   453
                        023A   454          .ENABL  LSB
                        C23A   455
                        023A   456 GETBYTE:
           53    D4     023A   457          CLRL    R3                      ; Return 0 if byte count = 0
           58    D5     023C   458          TSTL    R8                      ; Is byte count 0?
           2F    13     023E   459          BEQL    90$                     ; Yes
     53    8A    9A     0240   460          MOVZBL  (R10)+,R3               ; Get byte
           07    11     0243   461          BRB     10$                     ; Branch to common code
                        0245   462
                        0245   463
                        0245   464 PUTBYTE:
           58    D5     0245   465          TSTL    R8                      ; Is byte count 0?
           26    13     0247   466          BEQL    90$                     ; Yes
     8A    53    90     0249   467          MOVB    R3,(R10)+               ; Store byte
                        024C   468
                        024C   469
           58    D7     024C   470 10$:     DECL    R8                      ; Decr. byte count
           1F    13     024E   471          BEQL    90$                     ; Reached zero
5A   01FF  8F    B3     0250   472          BITW    #VA$M_BYTE,R10          ; Did address overflow onto new page?
           18    12     0255   473          BNEQ    90$                     ; No
           57    D6     0257   474          INCL    R7                      ; Yes, increment page number
                        0259   475
                        0259   476 ;
                        0259   477 ; Fall through to ...
                        0259   478 ;
                        0259   479
                        0259   480
                        0259   481 ;++
                        0259   482 ; DO_MAPPING - Subroutine to perform necessary mapping
                        0259   483 ;
                        0259   484 ; Inputs:
                        0259   485 ;       R6      Address of page table
                        0259   486 ;       R7      Page number of buffer
                        0259   487 ;       R10     Address to map
                        0259   488 ;       R11     Mapping switch:
                        0259   489 ;                       -1      Do physical -> virtual map
                        0259   490 ;                       0       No mapping required
                        0259   491 ;                       1       Do virtual -> physical translation
                        0259   492 ;
                        0259   493 ; Outputs:
                        0259   494 ;       R10     Address to use
                        0259   495 ;--
                        0259   496
                        0259   497 DO_MAPPING:
```

```
                     5B   D5  0259   498          TSTL    R11                          ; Any mapping required?
                     12   13  025B   499          BEQL    90$                          ; No
                     11   19  025D   500          BLSS    100$                         ; Yes, map physical to virtual
                 FFFFFE00 8F  CA  025F    .1       BICL    #^C<VA$M_BYTE>,R10           ; Yes, translate virtual to physical
                     5A       0265
-1                                   0266   502                                       ; Clear everything but byte offset
                   50   6647  D0  0266   503          MOVL    (R6)[R7],R0              ; Get PFN in R0
                15   09   50  F0  026A   504          INSV    R0,#VA$V_VPN,#PTE$S_PFN,R10  ; Insert PFN after byte offset
                     5A       026E
                          05  026F   505 90$:     RSB
                              0270   506
                              0270   507
                              0270   508 ;
                              0270   509 ; Map physical to virtual
                              0270   510 ;
                          00  0270   511 100$:    HALT                                 ; Not implemented yet
                              0271   512
                              0271   513          .DSABL  LSB
                              0271   514
                     00000271 0271    .1 CONSDD_DRVSIZ=.-CONSDD_START
-1                                   0271   516
                              0271   517          .END
```

```
$TABLE                          = 00000000 R   D  02
BTD$K_CONSOLE                   = 00000040     D
CLK$RE_INIT                       ********   X   03
CONSDD_DRIVER                     00000033 R   D  03
CONSDD_DRVSIZ                   = 00000271     D
CONSDD_START                      00000000 R   D  03
CPU_730                           00000017 R   D  03
CPU_750                           0000001D R   D  03
CPU_780                           00000016 R   D  03
CSRD_V_ERROR                    = 0000000F     D
CSRS_V_DONE                     = 00000007     D
CSTS_M_BREAK                    = 00000001     D
CSTS_V_READY                    = 00000007     D
DDNAME                            00000026 R   D  03
DD_CHECKCONTROLC                  0000017A R   D  03
DD_CNTRL                        = 00000002     D
DD_CONTINUE                     = 00000010     D
DD_DATA                         = 00000001     D
DD_ENDPKT                       = 00000040     D
DD_ERROR                          000001B6 R   D  03
DD_ERROR1                         0001E6 R   D  03
DD_ERROR2                         000148 R   D  03
DD_INIT                           000004     D
DD_READ                           00000002     D
DD_SELECT                         00000000 R   D  03
DD_TIMEOUT                        000001AF R   D  03
DD_WRITE                        = 00000003     D
DD_MAPPING                        00000259 R   D  03
DS$GL_FLAGS                       ********   X   03
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
END_OF_DATA                       00000143 R   D  03
EXE$GB_CPUTYPE                    ********   X   03
FUNC                            = 00000010     D
GETBYTE                           0000023A R   D  03
INIT                              00000024 R   D  03
IO$_READLBLK                    = 00000021     D
KB_POLL                           ********   X   03
MODE                            = 00000014     D
MRSP                              00000025 R   D  03
MRSP_SWITCH                     = 00000003     D
PR$_CSRD                        = 0000001D     D
PR$_CSRS                        = 0000001C     D
PR$_CSTD                        = 0000001F     D
PR$_CSTS                        = 0000001E     D
PR$_ICCS                        = 00000018     D
PR$_MAPEN                       = 00000038     D
PR$_SID_TYP780                  = 00000001     D
PTE$S_PFN                       = 00000015     D
PUTBYTE                           00000245 R   D  03
RECV_ONE_CHAR                     0000021D R   D  03
RECV_TWO_CHARS                    00000218 R   D  03
RECV_TWO_UPDSUM                   0000020A R   D  03
RESET_CLOCK                       0000019E R   D  03
SIZ...                          = 00000008     D
SS$_BUFBYTALI                   = 0000030C     D
SS$_CONTROLC                    = 00000651     D
SS$_CTRLERR                     = 00000054     D
SS$_NORMAL                      = 00000001     D
SS$_TIMEOUT                     = 0000022C     D
STKPTR                            00000020 R   D  03
VA$M_BYTE                       = 000001FF     D
VA$V_VPN                        = 00000009     D
WAIT_READY                        000001E8 R   D  03
```

```
XMIT_ONE_CHAR                   000001DD R  D  03
XMIT_TWO_CHARS                  000001D1 R  D  03
XMIT_TWO_UPDSUM                 000001CB R  D  03
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation		PSECT No.		Attributes								
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT NOVEC BYTE
ABS	00000000	(0.)	01	(1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
BOOTDRIVR_4	00000018	(24.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
BOOTDRIVR_2	00000271	(625.)	03	(3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES.
$TABLE	=00000000-R	121.3 (2)	121.3 (2)
BIT...	=00000008	45.5 (2)	45.5 (2)
BTD$K_CONSOLE	=00000040		121.3 (2)
CLK$RE_INIT	00000000-XR		355.23 (4)
CONSDD_DRIVER	00000033-R	200 (4)	121.3 (2)
CONSDD_DRVSIZ	=00000271	514.1 (4)	121.3 (2)
CONSDD_START	00000000-R	147.10 (3)	121.3 (2) 514.1 (4)
CPU_730	00000017-R	158.1 (3)	155 (3)
CPU_750	0000001D-R	158.4 (3)	155 (3)
CPU_780	00000016-R	156 (3)	155 (3)
CSRD_V_ERROR	=0000000F		#-420.9 (4)
CSRS_V_DONE	=00000007		#-388.28 (4)
CSTS_M_BREAK	=00000001		#-237 (4)
CSTS_V_READY	=00000007		#-420.7 (4)
DDNAME	00000026-R	198 (4)	121.3 (2)
DD_CHECKCONTROLC	0000017A-R	355.6 (4)	#-331.1 (4)
DD_CNTRL	=00000002		#-269 (4) #-340.2 (4)
DD_CONTINUE	=00000010		#-245 (4) #-292 (4) #-420.8 (4)
DD_DATA	=00000001		266 (4) #-318.2 (4)
DD_ENDPKT	=00000040		#-345 (4)
DD_ERROR	000001B6-R	355.33 (4)	#-331 (4) #-340.4 (4) #-346 (4) #-348 (4)
			#-355 (4) #-388.7 (4)
DD_ERROR1	000001E6-R	388.6 (4)	#-420.9 (4)
DD_ERROR2	0000014B-R	340.3 (4)	#-318.3 (4)
DD_INIT	=00000004		#-242 (4)
DD_READ	=00000002		265 (4) #-271 (4)
DD_SELECT	00000000-R	147.12 (3)	121.3 (2)
DD_TIMEOUT	000001AF-R	355.29 (4)	#-388.30 (4)
DD_WRITE	=00000003		265 (4)
DO_MAPPING	00000259-R	497 (4)	#-227 (4)
DS$GL_FLAGS	00000000-XR		355.9 (4)
DS$M_ABRTFLG	=00000040	45.5 (2)	
DS$M_BADTIME	=00100000	45.5 (2)	
DS$M_BATCH	=00400000	45.5 (2)	
DS$M_BRKCLR	=00001000	45.5 (2)	
DS$M_BRKPT	=00000800	45.5 (2)	
DS$M_CHARFLG	=00000100	45.5 (2)	
DS$M_CMDFLG	=00000080	45.5 (2)	
DS$M_CTRLC	=00000001	45.5 (2)	
DS$M_CTRLO	=00010000	45.5 (2)	
DS$M_DEVFLG	=00000200	45.5 (2)	
DS$M_DISABLCC	=01000000	45.5 (2)	
DS$M_DONFLG	=00002000	45.5 (2)	
DS$M_ERRFLG	=00000010	45.5 (2)	
DS$M_EXCEPT	=00080000	45.5 (2)	
DS$M_EXETST	=00040000	45.5 (2)	
DS$M_HLTFLG	=00000008	45.5 (2)	
DS$M_LODFLG	=00000002	45.5 (2)	
DS$M_MEMMGT	=00008000	45.5 (2)	
DS$M_OUTPUT	=00800000	45.5 (2)	

```
DS$M_RUBFLG                     =00000020     45.5    (2)
DS$M_SCRIPT                     =00200000     45.5    (2)
DS$M_SETIMR                     =02000000     45.5    (2)
DS$M_STRFLG                     =00000004     45.5    (2)
DS$M_SUBT                       =00004000     45.5    (2)
DS$M_SYSFLG                     =00000400     45.5    (2)
DS$M_TIMRON                     =00020000     45.5    (2)
DS$V_ABRTFLG                    =00000006     45.5    (2)
DS$V_BADTIME                    =00000014     45.5    (2)
DS$V_BATCH                      =00000016     45.5    (2)
DS$V_BRKCLR                     =0000000C     45.5    (2)
DS$V_BRKPT                      =0000000B     45.5    (2)
DS$V_CHARFLG                    =00000008     45.5    (2)
DS$V_CMDFLG                     =00000007     45.5    (2)
DS$V_CTRLC                      =00000C00     45.5    (2)    #-355.8  (4)
DS$V_CTRLO                      =00000010     45.5    (2)
DS$V_DEVFLG                     =00000009     45.5    (2)
DS$V_DISABLCC                   =00000018     45.5    (2)
DS$V_DONFLG                     =0000000D     45.5    (2)
DS$V_ERRFLG                     =00000004     45.5    (2)
DS$V_EXCEPT                     =00000013     45.5    (2)
DS$V_EXETST                     =00000012     45.5    (2)
DS$V_HLTFLG                     =00000003     45.5    (2)
DS$V_LODFLG                     =00000001     45.5    (2)
DS$V_MEMMGT                     =0000000F     45.5    (2)
DS$V_OUTPUT                     =00000017     45.5    (2)
DS$V_RUBFLG                     =00000005     45.5    (2)
DS$V_SCRIPT                     =00000015     45.5    (2)
DS$V_SETIMR                     =00000019     45.5    (2)
DS$V_STRFLG                     =00000002     45.5    (2)
DS$V_SUBT                       =0000000E     45.5    (2)
DS$V_SYSFLG                     =0000000A     45.5    (2)
DS$V_TIMRON                     =00000011     45.5    (2)
END_OF_DATA                      00000143-R   339     (4)    #-313     (4)
EXE$GB_CPUTYPE                   00000000-XR                 #-155     (3)
FUNC                            =00000010     188     (4)    #-272     (4)    #-285     (4)
GETBYTE                          0000023A-R   456     (4)    #-303     (4)    #-305     (4)
INIT                             00000024-R   197     (4)    #-227.5   (4)    #-247     (4)    #-355.10 (4)    #-355.36 (4)
IO$_READLBLK                    =00000021                   #-272     (4)    #-285     (4)
KB_POLL                          00000000-XR                  355.7   (4)
MODE                            =00000014     189     (4)    #-224     (4)
MRSP                             00000025-R   197.1   (4)    #-147.13  (3)    #-158.2   (3)    #-227.2  (4)    #-275.7  (4)
                                                            #-355.20  (4)    #-420.4   (4)
MRSP_SWITCH                     =00000003                   #-275.9   (4)
PR$_CSRD                        =0000001D                   #-241     (4)    #-420.3   (4)
PR$_CSRS                        =0000001C                   #-420.1   (4)
PR$_CSTD                        =0000001F                   #-388.3   (4)    #-420.8   (4)
PR$_CSTS                        =0000001F                   #-237     (4)    #-240     (4)    #-388.1  (4)    #-420.6  (4)
PR$_ICCS                        =00000016                   #-227.4   (4)
PR$_MAPEN                       =00000038                   #-222     (4)
PR$_SID_TYP780                  =00000001                   #-155     (3)
PTE$S_PFN                       =00000015                   #-504     (4)
PUTBYTE                          00000245-R   464     (4)    #-324     (4)    #-326     (4)
RECV_ONE_CHAR                    0000021D-R   420     (4)    #-244     (4)    #-291     (4)    #-416    (4)
RECV_TWO_CHARS                   00000218-R   415     (4)    #-327.1   (4)    #-349.3   (4)    #-409    (4)
RECV_TWO_UPDSUM                  0000020A-R   408     (4)    #-318.1   (4)    #-318.5   (4)    #-340.1  (4)    #-340.5  (4)
                                                            #-349.1   (4)
```

```
RESET_CLOCK                 0000019E-R   355.19 (4)    #-355.1  (4)    #-355.11 (4)    #-355.37 (4)
SIZ...                     =00000008     45.5   (2)      45.5   (2)
SS$_BUFBYTALI              =0000030C                    #-201.2  (4)
SS$_CONTROLC               =00000651                    #-355.14 (4)
SS$_CTRLERR                =00000054                    #-355.34 (4)
SS$_NORMAL                 =00000001                    #-355.2  (4)
SS$_TIMEOUT                =0000022C                    #-355.30 (4)
STKPTR                      00000020-R   196     (4)    #-201.5  (4)    #-355.12 (4)    #-361    (4)
VA$M_BYTE                  =000001FF                    #-472    (4)    #-500.1  (4)
VA$V_VPN                   =00000009                    #-504    (4)
WAIT_READY                  000001E8-R   388.23 (4)    #-388.2  (4)    #-420.2  (4)
XMIT_ONE_CHAR               000001DD-R   388     (4)    #-383.3  (4)
XMIT_TWO_CHARS              000001D1-R   383     (4)    #-239    (4)    #-243    (4)    #-284    (4)    #-306.4  (4)
XMIT_TWO_UPDSUM             000001CB-R   379     (4)    #-270    (4)    #-275    (4)    #-275.10 (4)    #-278    (4)
                                                       #-280    (4)    #-282    (4)    #-301    (4)    #-306.1  (4)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	119 (2)	119 (2)
$BTDDEF	1	47 (2)	47 (2)
$DEF	1	45.5 (2)	
$DEFINI	1	47 (2)	47 (2) 48 (2) 49 (2) 50 (2) 51 (2) 52 (2) 53 (2) 67 (2)
$EQU	1	45.5 (2)	
$GBLINI	2	45.5 (2)	45.5 (2)
$IODEF	17	48 (2)	48 (2)
$PRDEF	4	49 (2)	49 (2)
$PTEDEF	3	50 (2)	50 (2)
$RPBDEF	5	51 (2)	51 (2)
$SSDEF	21	52 (2)	52 (2)
$VADEF	1	53 (2)	53 (2)
$VIELD	1	45.5 (2)	45.5 (2)
$VIELD1	1	45.5 (2)	45.5 (2)
ASSUME	1	265 (4)	265 (4) 266 (4)
CASE	1	155 (3)	155 (3)
CPUDISP	1	152 (3)	152 (3)
DSFDEF	3	45.5 (2)	45.5 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.10	00:00:01.09
Command processing	141	00:00:00.75	00:00:03.09
Pass 1	688	00:00:14.71	00:00:24.71
Symbol table sort	0	00:00:01.81	00:00:02.09
Pass 2	180	00:00:04.14	00:00:09.08
Symbol table output	16	00:00:00.12	00:00:00.13
Psect synopsis output	5	00:00:00.02	00:00:00.04
Cross-reference output	31	00:00:00.54	00:00:01.13
Assembler run totals	1097	00:00:22.19	00:00:41.37

The working set limit was 1000 pages.
64820 bytes (127 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1134 non-local and 26 local symbols.
266 source lines were read in Pass 1, producing 0 object records in Pass 2.
55 pages of virtual memory were used to define 21 macros.

+--------------------------+
! Macro library statistics !
+--------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	0
DRB1:[DS.WORK]DS.MLB;218	3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	5
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	17

1190 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) CSDDBTDRV/UPDA=(CSDDBTDRV.UPD,CSDDBTDRV.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(2) 87 DECLARATIONS
(7) 298 SET BASE ADDRESS SUBROUTINE
(8) 341 SHOW BASE ADDRESS SUBROUTINE ; [13]
(9) 383 SET DEFAULTS SUBROUTINE
(11) 456 SHOW DEFAULTS SUBROUTINE
(13) 532 EXAMINE MEMORY AND REGISTERS SUBROUTINE
(15) 675 VRTYPEXAMINE Type out location EXAMINE'd
(20) 798 DEPOSIT MEMORY AND REGISTERS SUBROUTINE
(23) 943 SET BREAKPOINT SUBROUTINE
(25) 1001 CLEAR BREAKPOINT SUBROUTINE
(27) 1057 DSP$REMOVEBREAK Remove breakpoints
(28) 1107 SHOW BREAKPOINTS SUBROUTINE
(29) 1154 FIND_BPT FIND BPT IN BPT TABLE
(30) 1194 DEBUGGER CONDITION HANDLER
(31) 1241 BREAKPOINT CONDITION SUBROUTINE
(33) 1347 TBIT CONDITION SUBROUTINE
(35) 1405 TEMPORARY BREAKPOINT REMOVAL SUBROUTINE
(36) 1446 BREAKPOINT REPLACEMEMT SUBROUTINE
(37) 1499 FETCH_STORE COPY BYTE ROUTINE
(40) 1707 Handle EXAMINE/DEPOSIT of IPRs in KERNEL mode
(40) 1730 FETCH_STOR_PREG Move data to/from IPR
(40) 1775 DSV$DEBUG - Process DEBUG Command [14]

```
0000     1          .TITLE  DEBUG   EXAMINE, DEPOSIT, AND BREAKPOINT SUBROUTINES
0000     2          .IDENT  /07-15/
0000     3          .NLIST  CND
0000     4          .LIST   MEB
0000     5
0000     6 ;
0000     7 ; COPYRIGHT (C) 1977, 1980
0000     8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     9 ;
0000    10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    15 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    16 ; REMAIN IN DEC.
0000    17 ;
0000    18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    20 ; CORPORATION.
0000    21 ;
0000    22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    24
0000    25 ;++
0000    26 ; FACILITY:      VAX DIAGNOSTIC SUPERVISOR
0000    27 ;
0000    28 ; ABSTRACT:
0000    29 ;
0000    30 ; ENVIRONMENT:
0000    31 ;
0000    32 ; AUTHOR:        TOM SOUTTER     10-NOV-77       VERSION 01
0000    33 ;
0000    34 ; MODIFIED BY:
0000    35 ;
0000    36 ;                TOM SOUTTER     18-NOV-77       VERSION 02
0000    37 ;        01      DSPR #10 - LOCATE BREAKPOINTS VIA OPERATOR SPECIFIED BASE
0000    38 ;
0000    39 ;                KEN CHAPMAN     16-FEB-78       VERSION 03 (ESSAA-3.07).
0000    40 ;        02      REMOVED EXCEPTION HANDLER TO SCB.
0000    41 ;                Roger Riggs     18-JUL-78               (4.03)
0000    42 ;        03      Changed SS$_ARITH TO DS$_ARITH
0000    43 ;        04      Various enhancements and cleanup done,
0000    44 ;                'NEXT [n]' command.
0000    45 ;        05      Added code to restore opcode on hidden breakpoint
0000    46 ;        06      Modified FETCH_STORE to correctly report address on
0000    47 ;                COMET machine checks
0000    48 ;                Roger Riggs     29-NOV-1979
0000    49 ;        07      Added BSBW SCRIPT$STOP before call to CLI to halt script
0000    50 ;                mechanism on breakpoints and single step.
0000    51 ;                Dave Butenhof  16-JAN-80
0000    52 ;        08      Add code to examine/deposit internal processor registers.
0000    53 ;                Roger Riggs, 27-May-1980, Version 5.4
0000    54 ;        09      Changed radix on processor register output to hex.
0000    55 ;
0000    56 ;                Dave Butenhof, 30-sep-1980, Version 6.1
0000    57 ;        10      Make PSL typeout unaffected by /WORD or /BYTE.
```

```
                     0000     58 ;                Also change /ASCII to use !AF FAO directive, and also type
                     0000     59 ;                the longword in hex to aid in decoding non-printing strings.
                     0000     60 ;                Dave Butenhof, 23-feb-1981, version 6.3
                     0000     61 ;        11      fix truncation errors
                     0000     62 ;
                     0000     63 ;        12      - Jack Stansbury, 22-Oct-1981, Version 6.-
                     0000     64 ;                Fixed truncation errors. Also added .LIBRARY statements
                     0000     65 ;                for $DS and $DIAG.
                     0000     66 ;
                     0000     67 ;        13      - Jack Stansbury, 10-Nov-1981, Version 6.-
                     0000     68 ;                Added routines for VRSHOWBASE (to show the base address)
                     0000     69 ;                and VRSHOWDFLT (to show the default size and radix). Also
                     0000     70 ;                changed the error messages to mixed case.
                     0000     71 ;
                     0000     72 ;        14      - Richard Brown, 19-May-82, Version 6.8
                     0000     73 ;                Addition of routine to process the DEBUG user command.
                     0000     74 ;
                     0000     75 ;        15      - Domenic Andella, 17-Jul-84, Version 7.0
                     0000     76 ;                The 'DEP' command does a write followed by
                     0000     77 ;                read to display the data written. This causes
                     0000     78 ;                a mchk on 11/7xxx when depositing a one into
                     0000     79 ;                the Unibus Adapter Control Register (UACR),
                     0000     80 ;                and could cause similar problems in other
                     0000     81 ;                new CPU's. Problem solved by removing the
                     0000     82 ;                final examine from the 'DEPOSIT' command.
                     0000     83 ;                This makes VDS consistent with the DEPOSIT
                     0000     84 ;                command in XDELTA and DEBUG.
                     0000     85 ;--
```

```
                     0000     87          .SBTTL  DECLARATIONS
                     0000     88 ;
                     0000     89 ; INCLUDE FILES:
                     0000     90 ;
                     0000     91          .LIBRARY        /$DS/                   ;                                       [12]
                     0000     92          .LIBRARY        /$DIAG/                 ;                                       [12]
                     0000     93
                     0000     94 ;
                     0000     95 ; MACROS:
                     0000     96 ;
                     0000     97
                     0000     98 ;
                     0000     99 ; EQUATED SYMBOLS:
                     0000    100 ;
                     0000    101
                     0000    102
00000009             0000    103          HTAB      =   9                     ; ASCII HORIZONTAL TAB
00000003             0000    104          BPTCODE   == 03                     ; BREAKPOINT OPCODE
00000010             0000    105          PSLREG    =  16                     ; ERSATZ ADDRESS FOR 'PSL'
0000000F             0000    106          BPTMAX    == 15                     ; MAXIMUM NUMBER OF BREAKPOINTS
00000020             0000    107          EXAMSIZE  =  32                     ; SIZE OF DISPLAY BUFFER FOR EXAMINE
00000050             0000    108          PSLBUFSZ  =  80                     ; SIZE OF PSL MNEMONICS BUFFER
00000000             0000    109          INITBASE  =   0                     ; INITIAL BASE ADDRESS FOR EXAM/DEP
00000004             0000    110          INITSIZE  =   4                     ; INITIAL BYTE COUNT FOR EXAM/DEP
                     0000    111
00000001             0000    112          SS$_NORMAL      = 1                 ; SUCCESSFUL COMPLETION
                     0000    113
00010E00             0000    114          DEBUGGER_SIZE = 135 * 512           ;NO. OF PAGES TIMES BYTES PER PAGE[14]
                     0000    115
                     0000    119          CLIDEF                              ; CLI COMMAND BLOCK OFFSETS
                     0000    120          DSFDEF                              ; CONTROL FLAG DEFINITIONS
                     0000    121          $DS_DSDEF                           ; SUPERVISOR STATUS CODES
                     0000    122          $PSLDEF                             ; PROCESSOR STATUS BITS
                     0000    123          $PRDEF                              ; PROCESSOR REGISTERS
                     0000    124          $SETPRTDEF                          ; SETPRT arglist offsets
                     0000    125          $DS_DSADEF                          ; Define DSA$x codes
                     0000    126          CMKDEF                              ; Define CHMK codes
```

```
                                0000   131 ; GLOBAL REFERENCES                                                          [14]
                                0000   132
                                0000   133         .GLOBAL DIAG_FILE_NAME          ;FILENAME OF LAST DIAG. LOADED  [14]
                                0000   134
                                0000   135 ;
                                0000   136 ; OWN STORAGE:
                                0000   137 ;
                                0000   138
                            00000000   139         .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                0000   140
                                0000   141 DEBUG$GB_FLAGS::
                                  00   0000   142         .BYTE   0
                                       0001   143                 T=0                     ; T-BIT TRAP EXPECTED
                                       0001   144                         M_TBIT=1 @ V_TBIT
                                       0001   145                 V_STEP=1                ; STEP EXECUTION, SET T-BIT AFTER T-BIT TRAP
                                       0001   146                         M_STEP=1 @ V_STEP
                                       0001   147 DS$AB_BPTINST::
00'00'00'00'00'00'00'00'00'00'00'00'00'  0001   148         .BYTE   0[BPTMAX+1]             ; SAVED INSTRUCTION TABLE
                     00'00'00'00'00'00'  000D
                                       0011   149         .ALIGN  LONG
                                       0014   150 DS$AA_BPTADDR::
00000000'00000000'00000000'00000000'00000000'  0014   151         .LONG   0[BPTMAX+1]             ; BREAKPOINT LOCATION TABLE
00000000'00000000'00000000'00000000'00000000'  0024
00000000'00000000'00000000'00000000'00000000'  0034
00000000'00000000'00000000'00000000'00000000'  0044
                                       0054   152 DFLTSIZE:
                                00000004   0054   153         .LONG   INITSIZE                ; EXAMINE/DEPOSIT SIZE QUALIFIER
                                       0058   154
                                       0058   155 BASEADDRESS:
                                00000000   0058   156         .LONG   INITBASE                ; ADDRESS BIAS FOR EXAMINE/DEPOSIT
                                       005C   157
                                00000064   005C   158 NEW_AP_FP:      .BLKQ   1               ; SPACE FOR AP/FP
                                00000068   0064   159 NEW_SP:         .BLKL   1               ; OPERATOR MODIFIED 'SP'
                                00000070   0068   160 NEW_PC_PSL:     .BLKQ   1               ; OPERATOR MODIFIED 'PC/PSL'
                                       0070   161
                    00000078'00000003'  0070   162 FMTDIRECTIVE:   .LONG   20$-10$,10$      ; QUADWORD STRING DESCRIPTOR
                                3F 3F 21   0078   163 10$:     .ASCII  .!??.                   ; FORMAT DIRECTIVE STRING
                                       007B   164 20$:
                                       007B   165
                                       007B   166 EXAMQWDBUF:
                    00000083'00000020  007B   167         .LONG   EXAMSIZE,EXAMBUFFER     ; QUADWORD BUFFER DESCRIPTOR
                                       0083   168 EXAMBUFFER:
                                000000A3   0083   169         .BLKB   EXAMSIZE                ; DISPLAY BUFFER FOR EXAMINE
                                       00A3   170 PSLBUFFER:
                                000000F3   00A3   171         .BLKB   PSLBUFSZ                ; BUFFER FOR PSL MNEMONICS
                                       00F3   172
                                       00F3   173 FAOLST:
                                       00F3   174         $FAOL   FMTDIRECTIVE,           - ; CONTROL STRING POINTER
                                       00F3   175                 EXAMQWDBUF+2,        - ; RETURNED STRING LENGTH (WORD)
                                       00F3   176                 EXAMQWDBUF,          - ; RETURN STRING BUFFER DESCRIPTOR
                                       00F3   177                 FAOARG                 ; ARGLIST POINTER (UNCOUNTED)
                                00000004'  00F3                   .ADDRESS        4
                                00000070'  00F7                   .ADDRESS        FMTDIRECTIVE
                                0000007D'  00FB                   .ADDRESS        EXAMQWDBUF+2
                                0000007B'  00FF                   .ADDRESS        EXAMQWDBUF
                                00000107'  0103                   .ADDRESS        FAOARG
                                       0107   178
```

```
                                        0107   179 FAOARG:
                              0000010B  0107   180         .BLKL   1                       ; FAO OPTIONAL ARGS LIST
                                        010B   181
                                        010B   182
                                        010B   183 ; The following storage allocation is for use by the routine which      [14]
                                        010B   184 ; loads the debugger.                                                   [14]
                                        010B   185
62 65 64 20 67 6E 69 64 61 6F 4C 00'    010B   186 FMT_DBG_MSG:    .ASCIC "Loading debugger...!/"          ;                [14]
      2F 21 2E 2E 2E 72 65 67 67 75     0117
                                 15     010B
6F 6C 20 72 65 67 67 75 62 65 44 00'    0121   187 FMT_DBG_LOAD:   .ASCIC "Debugger loaded at !XL!/"       ;                [14]
21 4C 58 21 20 74 61 20 64 65 64 61     012D
                                 2F     0139
                                 18     0121
4F 4C 20 54 27 4E 41 43 20 3F 3F 00'    C13A   188 FMT_NO_ROOM:    .ASCIC "?? CAN'T LOAD DEBUGGER - NOT ENOUGH FREE MEMORY.!/";[14]
20 52 45 47 47 55 42 45 44 20 44 41     0146
48 47 55 4F 4E 45 20 54 4F 4E 20 2D     0152
59 52 4F 4D 45 4D 20 45 45 52 46 20     015E
                        2F 21 2E        016A
                              32        013A
                    00000000 00000000   016D   189 DBG_ADDR_REQ:   .QUAD 0                 ;ARRAY TO REQUEST ADDR.          [14]
                                        0175   190                                         ; RANGE                          [14]
                    0000000C 00000000   0175   191 DBG_ADDR_RTN:   .QUAD 0                 ;ARRAY TO RECEIVED ACTUAL ARRAY  [14]
                                        017D   192                                         ; ALLOCATED                      [14]
                              00000000  017D   193 DEBUG_LO::      .LONG 0                 ;LOWEST ADDR. ALLOCATED TO DEBUGGER[14]
                              00000000  0181   194 DEBUG_HI::      .LONG 0                 ;HIGHEST ADDR. ALLOCATED TO DEBUGGER[14]
                              00000000  0185   195 SYMTAB_HI::     .LONG 0                 ;TOP OF SYMBOL TABLE AREA        [14]
```

```
                                00000000   197          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                    0000   198
                                    0000   199          MODNAM  DEBUG                   ; SUPERVISOR MODULE IDENTIFIER
                   47 55 42 45 44 00' 0000      $MODULE:        .ASCIC  \DEBUG\
                                    05 0000
                                    03 0006   200 BPT_LITERAL:   .BYTE   BPTCODE         ; A BPT INSTRUCTION
                                    0007   201
                                    0007   202 FMT_BASE_OUT:                                                ;       [13]
74 6E 65 72 72 75 63 20 65 68 54 00' 0007   203          .ASCIC  "The current base address is !XL(X).!/"     ;       [13]
73 65 72 64 64 61 20 65 73 61 62 20  0013
2E 29 58 28 4C 58 21 20 73 69 20 73  001F
                        2F 21        002B
                                    25 0007
                                    002D   204 FMT_DFLT_OUT:                                                ;       [13]
64 61 72 20 74 6E 65 72 72 75 43 00' 002D   205          .ASCIC  "Current radix: !AC.!/Current size:  !AC.!/" ;       [13]
75 43 2F 21 2E 43 41 21 20 3A 78 69  0039
20 3A 65 7A 69 73 20 74 6E 65 72 72  0045
                  2F 21 2E 43 41 21  0051
                                    2A 002D
                                    0058   206 FMT_HEX:                                                    ;       [13]
6C 61 6D 69 63 65 64 61 78 65 48 00' 0058   207          .ASCIC  "Hexadecimal"                               ;       [13]
                                    08 0058
                                    0064   208 FMT_DEC:                                                    ;       [13]
                6C 61 6D 69 63 65 44 00' 0064   209          .ASCIC  "Decimal"                               ;       [13]
                                    07 0064
                                    006C   210 FMT_OCT:                                                    ;       [13]
                   6C 61 74 63 4F 00' 006C   211          .ASCIC  "Octal"                                 ;       [13]
                                    05 006C
                                    0072   212 FMT_LONG:                                                   ;       [13]
            64 72 6F 77 67 6E 6F 4C 00' 0072   213          .ASCIC  "Longword"                              ;       [13]
                                    08 0072
                                    007B   214 FMT_WORD:                                                   ;       [13]
                      64 72 6F 57 00' 007B   215          .ASCIC  "Word"                                  ;       [13]
                                    04 007B
                                    0080   216 FMT_BYTE:                                                   ;       [13]
                      65 74 79 42 00' 0080   217          .ASCIC  "Byte"                                  ;       [13]
                                    04 0080
                                    0085   218 FMTBPT:
6E 69 6F 70 6B 61 65 72 42 2F 21 00' 0085   219          .ASCIC  "!/Breakpoint at: !XL!/"                    ;       [13]
   2F 21 4C 58 21 20 3A 74 61 20 74  0091
                                    16 0085
                                    009C   220 FMTSTEP:
   2F 21 4C 58 21 20 3A 4C 58 21 00' 009C   221          .ASCIC  "!XL: !XL!/"
                                    0A 009C
                                    00A7   222 FMTBPTADR:
   2F 21 29 58 28 4C 58 21 5F 21 00' 00A7   223          .ASCIC  .!_!XL(X)!/.
                                    0A 00A7
                                    00B2   224 FMTBPTLST:
65 72 62 20 74 6E 65 72 72 75 43 00' 00B2   225          .ASCIC  .Current breakpoints:!/.                    ;       [13]
   2F 21 3A 73 74 6E 69 6F 70 6B 61  00BE
                                    16 00B2
                                    00C9   226 FMTBPTNONE:
6E 65 72 72 75 63 20 65 6E 6F 4E 00' 00C9   227          .ASCIC  .None currently set!/.                      ;       [13]
         2F 21 74 65 73 20 79 6C 74  00D5
                                    14 00C9
                                    00DE   228 FMTBPTNOTSET:
20 74 6E 69 6F 70 6B 61 65 72 42 00' 00DE   229          .ASCIC  .Breakpoint was not set!!!/.                ;       [13]
```

```
21 74 65 73 20 74 6F 6E 20 73 61 77  00EA
                        2F 21 21     00F6
                              1A     00DE
                                     00F9   230 FMTINVLDREG:
20 64 69 6C 61 76 6E 49 20 3F 3F 00' 00F9   231         .ASCIC  .?? Invalid register mnemonic!/.                    ;       [13]
65 6E 6D 20 72 65 74 73 69 67 65 72  0105
                  2F 21 63 69 6E 6F  0111
                              1E     00F9
                                     0118   232 FMTEXAM_ASCII:
21 22 46 41 21 22 00000120'010E0000' 0118   233         .ASCID  .''!AF''!_!#XL.
                        4C 58 23 21 5F 0126
                                     012B   234 FMTEXAM:
43 41 21 5F 21 53 41 21 20 20 20 00' 012B   235         .ASCIC  ''   !AS!_!AC!/''
                              2F 21  0137
                                 0D  C12B
                                     0139   236 FMTNOBPTSLEFT:
65 72 62 20 4C 55 21 20 6C 6C 41 00' 0139   237         .ASCIC  .All !UL breakpoints currently in use!!!/.          ;       [13]
72 75 63 20 73 74 6E 69 6F 70 6B 61  0145
73 75 20 6E 69 20 79 6C 74 6E 65 72  0151
                        2F 21 21 21  015D
                              28     0139
```

```
6F 70 73 65 72 20 6F 4E 20 3F 3F 00' 0162   239 T_MACHCK:       .ASCIC  '?? No response/invalid operation'                ;       [13]
20 64 69 6C 61 76 6E 69 2F 65 73 6E  016E
            6E 6F 69 74 61 72 65 70 6F  017A
                                    20  0162
76 20 73 73 65 63 63 41 20 3F 3F 00' 0183   240 T_ACCVIO:       .ASCIC  '?? Access violation'                            ;       [13]
               6E 6F 69 74 61 6C 6F 69  018F
                                    13  0183
74 61 6C 73 6E 61 72 54 20 3F 3F 00' 0197   241 T_TRANSL:       .ASCIC  '?? Translation not valid'                       ;       [13]
69 6C 61 76 20 74 6F 6E 20 6E 6F 69  01A3
                                    64  01AF
                                    18  0197
20 67 6E 69 72 75 64 20 43 41 21 00' 01B0   242 FMT_EXCEPTION:  .ASCIC  ''!AC during reference to X'!XL'!/''              ;       [13]
6F 74 20 65 63 6E 65 72 65 66 65 72  01BC
            2F 21 27 4C 58 21 27 58 20  01C8
                                    20  C1B0
      6D 6F 72 66 20 64 61 65 72 00' 01D1   243 T_PREGREAD:     .ASCIC  ''read from''                                    ;       [13]
                                    09  01D1
         6F 74 20 65 74 69 72 77 00' 01DB   244 T_PREGWRITE:    .ASCIC  ''write to''                                     ;       [13]
                                    08  01DB
41 21 20 74 27 6E 61 43 20 3F 3F 00' 01E4   245 FMT_BADPREG:    .ASCIC  ''?? Can't !AC P!XB!/''                          ;       [13]
               2F 21 42 58 21 50 20 43  01F0
                                    13  01E4
                                        01F8   246 APSLMNE::
44 50 46 2C 2C 2C 50 54 2C 4D 43 00' 01F8   247             .ASCIC  \CM,TP,,,FPD,IS,CUR=2@,PRV=2@,,IPL=5^X,\ -
50 2C 40 32 3D 52 55 43 2C 53 49 2C  0204
35 3D 4C 50 49 2C 2C 40 32 3D 56 52  0210
44 2C 2C 2C 2C 2C 2C 2C 2C 2C 58 5E  021C
2C 4E 2C 54 2C 56 49 2C 55 46 2C 56  0228
                  43 2C 56 2C 5A     0234
                              40     01F8
                                     0239   248                     \,,,,,,,,DV,FU,IV,T,N,Z,V,C\
                                     0239   249
                           00000004  0239   250 MODELST::  .LONG          4
                           0000024D' 023D   251           .ADDRESS       AKM
                           00000254' 0241   252           .ADDRESS       AEM
                           0000025E' 0245   253           .ADDRESS       ASM
                           00000269' 0249   254           .ADDRESS       AUM
                                     024D   255
               4C 45 4E 52 45 4B 00' 024D   256 AKM:     .ASCIC  .KERNEL.
                                 06  024D
      45 56 49 54 55 43 45 58 45 00' 0254   257 AEM:     .ASCIC  .EXECUTIVE.
                                 09  0254
   52 4F 53 49 56 52 45 50 55 53 00' 025E   258 ASM:     .ASCIC  .SUPERVISOR.
                                 0A  025E
                  52 45 53 55 00' 0269   259 AUM:     .ASCIC  .USER.
                              04  0269
```

```
                    026E    261 REGNAMLST:
000002B2' 026E      262             .ADDRESS  AR0
000002B5' 0272      263             .ADDRESS  AR1
000002B8' 0276      264             .ADDRESS  AR2
000002BB' 027A      265             .ADDRESS  AR3
000002BE' 027E      266             .ADDRESS  AR4
000002C1' 0282      267             .ADDRESS  AR5
000002C4' 0286      268             .ADDRESS  AR6
000002C7' 028A      269             .ADDRESS  AR7
000002CA' 028E      270             .ADDRESS  AR8
000002CD' 0292      271             .ADDRESS  AR9
000002D0' 0296      272             .ADDRESS  AR10
000002D4' 029A      273             .ADDRESS  AR11
000002D8' 029E      274             .ADDRESS  AAP
000002DB' C2A2      275             .ADDRESS  AFP
000002DE' 02A6      276             .ADDRESS  ASP
000002E1' 02AA      277             .ADDRESS  APC
000002E4' 02AE      278             .ADDRESS  APSL
                    02B2    279
30 52 00' 02B2      280 AR0:        .ASCIC  .R0.
      02  02B2
31 52 00' 02B5      281 AR1:        .ASCIC  .R1.
      02  02B5
32 52 00' 02B8      282 AR2:        .ASCIC  .R2.
      02  02B8
33 52 00' 02BB      283 AR3:        .ASCIC  .R3.
      02  02BB
34 52 00' 02BE      284 AR4:        .ASCIC  .R4.
      02  02BE
35 52 00' 02C1      285 AR5:        .ASCIC  .R5.
      02  02C1
36 52 00' 02C4      286 AR6:        .ASCIC  .R6.
      02  02C4
37 52 00' 02C7      287 AR7:        .ASCIC  .R7.
      02  02C7
38 52 00' 02CA      288 AR8:        .ASCIC  .R8.
      02  02CA
39 52 00' 02CD      289 AR9:        .ASCIC  .R9.
      02  02CD
30 31 52 00' 02D0   290 AR10:       .ASCIC  .R10.
      03  02D0
31 31 52 00' 02D4   291 AR11:       .ASCIC  .R11.
      03  02D4
50 41 00' 02D8      292 AAP:        .ASCIC  .AP.
      02  02D8
50 46 00' 02DB      293 AFP:        .ASCIC  .FP.
      02  02DB
50 53 00' 02DE      294 ASP:        .ASCIC  .SP.
      02  02DE
43 50 00' 02E1      295 APC:        .ASCIC  .PC.
      02  02E1
4C 53 50 00' 02E4   296 APSL:       .ASCIC  .PSL.
      03  02E4
```

```
                     02E8   298          .SBTTL  SET BASE ADDRESS SUBROUTINE
                    00000000 299         .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                        0000 300 ;++
                        0000 301 ; FUNCTIONAL DESCRIPTION:
                        0000 302 ;
                        0000 303 ;       SETS UP AN ADDRESS BIAS VALUE TO BE USED BY SUBSEQUENT
                        0000 304 ;       EXAMINE AND DEPOSIT OPERATOR COMMANDS
                        0000 305 ;
                        0000 306 ; CALLING SEQUENCE:
                        0000 307 ;
                        0000 308 ;       BSBW    VRSETBASE
                        0000 309 ;
                        0000 310 ; INPUT PARAMETERS:
                        0000 311 ;
                        0000 312 ;       R2      = BASE OF CLI DATA FRAME.
                        0000 313 ;       CLI$L_ADDRESS(R2)       ; OPERATOR SPECIFIED BIAS
                        0000 314 ;
                        0000 315 ; IMPLICIT INPUTS:
                        0000 316 ;
                        0000 317 ;       NONE
                        0000 318 ;
                        0000 319 ; OUTPUT PARAMETERS:
                        0000 320 ;
                        0000 321 ;       BASEADDRESS                     ; RECEIVES BIAS VALUE
                        0000 322 ;
                        0000 323 ; IMPLICIT OUTPUTS:
                        0000 324 ;
                        0000 325 ;       NONE
                        0000 326 ;
                        0000 327 ; COMPLETION CODES:
                        0000 328 ;
                        0000 329 ;       NONE
                        0000 330 ;
                        0000 331 ; SIDE EFFECTS:
                        0000 332 ;
                        0000 333 ;       NONE
                        0000 334 ;
                        0000 335 ;--
                        0000 336
                        0000 337 VRSETBASE::                    ;
00000058'EF   18 A2  D0 0000 338         MOVL    CLI$L_ADDRESS(R2),L^BASEADDRESS ; STORE THE SPECIFIED BIAS      [12]
                     05 0008 339         RSB                             ; RETURN
```

```
                                0009   341          .SBTTL  SHOW BASE ADDRESS SUBROUTINE                                ;       [13]
                                0009   342 ;++
                                0009   343 ; FUNCTIONAL DESCRIPTION:
                                0009   344 ;
                                0009   345 ;       Shows the current base address.
                                0009   346 ;
                                0009   347 ; CALLING SEQUENCE:
                                0009   348 ;
                                0009   349 ;       BSBW    VRSHOWBASE
                                0009   350 ;       JSB     VRSHOWBASE
                                0009   351 ;
                                0009   352 ; INPUT PARAMETERS:
                                0009   353 ;
                                0009   354 ;       NONE
                                0009   355 ;
                                0009   356 ; IMPLICIT INPUTS:
                                0009   357 ;
                                0009   358 ;       BASEADDRESS : The current base address
                                0009   359 ;
                                0009   360 ; OUTPUT PARAMETERS:
                                0009   361 ;
                                0009   362 ;       NONE
                                0009   363 ;
                                0009   364 ; IMPLICIT OUTPUTS:
                                0009   365 ;
                                0009   366 ;       NONE
                                0009   367 ;
                                0009   368 ; COMPLETION CODES:
                                0009   369 ;
                                0009   370 ;       NONE
                                0009   371 ;
                                0009   372 ; SIDE EFFECTS:
                                0009   373 ;
                                0009   374 ;       NONE
                                0009   375 ;
                                0009   376 ;--
                                0009   377
                                0009   378 VRSHOWBASE::                                                          ;       [13]
                                0009   379         $DS_PRINTF_S    L^FMT_BASE_OUT, -                           ; Print the base address        [13]
                                0009   380                           L^BASEADDRESS                             ;                               [13]
      00000058'EF    DD         0009                       PUSHL   L^BASEADDRESS
      00000007'EF    9F         000F                       PUSHAB  L^FMT_BASE_OUT
00000000'9F    02    FB         0015                       CALLS   #$$N, @#DS$PRINTF
                     05         001C   381         RSB                                                             ; Return to caller              [13]
```

```
001D   383          .SBTTL  SET DEFAULTS SUBROUTINE
001D   384 ;++
001D   385 ; FUNCTIONAL DESCRIPTION:
001D   386 ;
001D   387 ;       SETS UP DEFAULT PARAMETERS TO BE USED BY SUBSEQUENT
001D   388 ;       EXAMINE AND DEPOSIT OPERATOR COMMANDS
001D   389 ;
001D   390 ; CALLING SEQUENCE:
001D   391 ;
001D   392 ;       BSBW    VRSETDFLT
001D   393 ;
001D   394 ; INPUT PARAMETERS:
001D   395 ;
001D   396 ;       R2      = BASE OF CLI DATA FRAME.
C01D   397 ;       CLI$L_FLAGS(R2) ; OPERATOR SPECIFIED DEFAULT(S)
001D   398 ;
001D   399 ; IMPLICIT INPUTS:
001D   400 ;
001D   401 ;       NONE
001D   402 ;
001D   403 ; OUTPUT PARAMETERS:
001D   404 ;
001D   405 ;       DS$GL_RADIX                     ; RECEIVES RADIX VALUE
001D   406 ;       DFLTSIZE                        ; RECEIVES SIZE VALUE IN BYTES
001D   407 ;
001D   408 ; IMPLICIT OUTPUTS:
001D   409 ;
001D   410 ;       NONE
001D   411 ;
001D   412 ; COMPLETION CODES:
001D   413 ;
001D   414 ;       NONE
001D   415 ;
001D   416 ; SIDE EFFECTS:
001D   417 ;
001D   418 ;       NONE
001D   419 ;
001D   420 ;--
```

```
                                001D   422 VRSETDFLT::                             ;
                                001D   423
                                001D   424 ;+
                                001D   425 ; FIRST, CHECK RADIX FLAGS AND SET DEFAULT TO SPECIFIED VALUE
                                001D   426 ; (IF MORE THAN ONE SPECIFIED, USE MAXIMIZED VALUE)
                                001D   427 ;-
                                001D   428
          07 62    11    E1     001D   429          BBC     #CLI$V_OCT,CLI$L_FLAGS(R2),10$ ; OCTAL ?
00000000'EF    08    D0        0021   430          MOVL    #8,L^DS$GL_RADIX            ; YES, SET DEFAULT                  [12]
          07 62    10    E1     0028   431 10$:     BBC     #CLI$V_DEC,CLI$L_FLAGS(R2),20$ ; DECIMAL ?                          [12]
00000000'EF    0A    D0        002C   432          MOVL    #10,L^DS$GL_RADIX           ; YES, SET DEFAULT                  [12]
          07 62    12    E1     0033   433 20$:     BBC     #CLI$V_HEX,CLI$L_FLAGS(R2),30$ ; HEX ?                              [12]
00000000'EF    10    D0        0037   434          MOVL    #16,L^DS$GL_RADIX           ; YES, SET DEFAULT                  [12]
                                003E   435 30$:
                                C03E   436
                                003E   437 ;+
                                003E   438 ; HERE, CHECK SIZE FLAGS AND SET DEFAULT TO SPECIFIED NUMBER OF
                                003E   439 ; BITS  (IF MORE THAN ONE SPECIFIED, USE MAXIMIZED VALUE)
                                003E   440 ;-
                                003E   441
          07 62    0D    E1     003E   442          BBC     #CLI$V_BYTE,CLI$L_FLAGS(R2),40$ ; BYTE ?
00000054'EF    01    D0        0042   443          MOVL    #1,L^DFLTSIZE               ; YES, SET SIZE IN BYTES            [12]
          07 62    0E    E1     0049   444 40$:     BBC     #CLI$V_WORD,CLI$L_FLAGS(R2),50$ ; WORD ?                            [12]
00000054'EF    02    D0        004D   445          MOVL    #2,L^DFLTSIZE               ; YES, SET SIZE IN BYTES            [12]
          07 62    0F    E1     0054   446 50$:     BBC     #CLI$V_LONG,CLI$L_FLAGS(R2),60$ ; LONGWORD ?                        [12]
00000054'EF    04    D0        0058   447          MOVL    #4,L^DFLTSIZE               ; YES, SET SIZE IN BYTES            [12]
                                005F   448 60$:
                                005F   449
                                005F   450 ;+
                                005F   451 ; RETURN
                                005F   452 ;-
                                005F   453
                          05    005F   454          RSB                                 ; RETURN
```

```
0060    456          .SBTTL  SHOW DEFAULTS SUBROUTINE
0060    457 ;++
0060    458 ; FUNCTIONAL DESCRIPTION:
0060    459 ;
0060    460 ;       Show the current default size and radix.
0060    461 ;
0060    462 ; CALLING SEQUENCE:
0060    463 ;
0060    464 ;       BSBW    VRSHOWDFLT
0060    465 ;       JSB     VRSHOWDFLT
0060    466 ;
0060    467 ; INPUT PARAMETERS:
0060    468 ;
0060    469 ;       NONE
C060    470 ;
0060    471 ; IMPLICIT INPUTS:
0060    472 ;
0060    473 ;       NONE
0060    474 ;
0060    475 ; OUTPUT PARAMETERS:
0060    476 ;
0060    477 ;       NONE
0060    478 ;
0060    479 ; IMPLICIT OUTPUTS:
0060    480 ;
0060    481 ;       NONE
0060    482 ;
0060    483 ; COMPLETION CODES:
0060    484 ;
0060    485 ;       NONE
0060    486 ;
0060    487 ; SIDE EFFECTS:
0060    488 ;
0060    489 ;       NONE
0060    490 ;
0060    491 ;--
```

```
                                0060   493 VRSHOWDFLT::                                ;                                                [13]
                                0060   494                                             ;                                                [13]
                                0060   495         ; First, save R0, R1, and R2.       ;                                                [13]
                                0060   496         ;                                   ;                                                [13]
                          07    BB    0060   497         PUSHR   #^M<R0,R1,R2>           ;                                                [13]
                                0062   498         ;                                   ;                                                [13]
                                0062   499         ; Put the FMT for the radix in R0,  ;                                                [13]
                                0062   500         ; the FMT for the size in R1.       ;                                                [13]
                                0062   501         ;                                   ;                                                [13]
          52    00000000'EF   D0    0062   502         MOVL    L^DS$GL_RADIX, R2       ; Store the radix in R2                          [13]
                     52   10    D1    0069   503         CMPL    #16, R2                 ; Hexadecimal?                                   [13]
                          09    12    006C   504         BNEQ    10$                     ; If not, branch                                 [13]
          50    00000058'EF   9E    006E   505         MOVAB   L^FMT_HEX, R0           ; Stor  format for hex                           [13]
                          15    11    0075   506         BRB     30$                     ;                                                [13]
                     52   0A    D1    0077   507 10$:    CMPL    #10, R2                 ; Decimal ?                                      [13]
                          09    12    007A   508         BNEQ    20$                     ; If not, branch                                 [13]
          50    00000064'EF   9E    007C   509         MOVAB   L^FMT_DEC, R0           ; Store format for decimal                       [13]
                          07    11    0083   510         BRB     30$                     ;                                                [13]
          50    0000006C'EF   9E    0085   511 20$:    MOVAB   L^FMT_OCT, R0           ; Must be octal                                  [13]
                                008C   512 30$:                                         ;                                                [13]
          52    00000054'EF   D0    008C   513         MOVL    L^DFLTSIZE, R2          ; Store the size in R2                           [13]
                     52   04    D1    0093   514         CMPL    #4, R2                  ; Longword?                                      [13]
                          09    12    0096   515         BNEQ    40$                     ; If not, branch                                 [13]
          51    00000072'EF   9E    0098   516         MOVAB   L^FMT_LONG, R1          ; Store format for long                          [13]
                          15    11    009F   517         BRB     60$                     ;                                                [13]
                     52   02    D1    00A1   518 40$:    CMPL    #2, R2                  ; Word?                                          [13]
                          09    12    00A4   519         BNEQ    50$                     ; If not, branch                                 [13]
          51    0000007B'EF   9E    00A6   520         MOVAB   L^FMT_WORD, R1          ; Store format for word                          [13]
                          07    11    00AD   521         BRB     60$                     ;                                                [13]
          51    00000080'EF   9E    00AF   522 50$:    MOVAB   L^FMT_BYTE, R1          ; Store format for byte                          [13]
                                00B6   523 60$:                                         ;                                                [13]
                                00B6   524                                             ;                                                [13]
                                00B6   525         ; Print the size and radix.         ;                                                [13]
                                00B6   526         ;                                   ;                                                [13]
                                00B6   527         $DS_PRINTF_S    L^FMT_DFLT_OUT, -       ; Print in this format                   [13]
                                00B6   528                           R0, R1             ; Size and radix                                 [13]
                          51    DD    00B6                 PUSHL   R1
                          50    DD    00B8                 PUSHL   R0
               0000002D'EF      9F    00BA                 PUSHAB  L^FMT_DFLT_OUT
          00000000'9F   03      FB    00C0                 CALLS   #$$N, @#DS$PRINTF
                          07    BA    00C7   529         POPR    #^M<R0,R1,R2>           ; Restore registers                              [13]
                          05    00C9   530         RSB                                 ; Return to caller                               [13]
```

```
                        532          .SBTTL  EXAMINE MEMORY AND REGISTERS SUBROUTINE
                        533 ;++
                        534 ; FUNCTIONAL DESCRIPTION:
                        535 ;
                        536 ;       LOCATES AND DISPLAYS SPECIFIED MEMORY OR GENERAL REGISTERS
                        537 ;       TO THE OPERATOR IN THE SPECIFIED RADIX OR ASCII.
                        538 ;       ALL SUPERVISOR OVERHEAD (I.E., STACK & REGISTER USAGE)
                        539 ;       IS TRANSPARENT TO THE OPERATOR.
                        540 ;
                        541 ; CALLING SEQUENCE:
                        542 ;
                        543 ;       BSBW    VREXAMINE
                        544 ;
                        545 ; INPUT PARAMETERS:
                        546 ;
                        547 ;       R2      = BASE OF CLI DATA FRAME.
                        548 ;       CLI$L_FLAGS(R2)         ; RADIX/SIZE SPECIFIERS
                        549 ;       CLI$L_ADDRESS(R2)       ; ADDRESS SPECIFIER
                        550 ;
                        551 ; IMPLICIT INPUTS:
                        552 ;
                        553 ;       (AP) = BASE ADDRESS OF PROGRAM REGISTER CONTEXT
                        554 ;       DS$GL_RADIX                     ; DEFAULT RADIX
                        555 ;       DFLTSIZE                        ; DEFAULT SIZE IN BYTES
                        556 ;
                        557 ; OUTPUT PARAMETERS:
                        558 ;
                        559 ;       NONE
                        560 ;
                        561 ; IMPLICIT OUTPUTS:
                        562 ;
                        563 ;       NONE
                        564 ;
                        565 ; COMPLETION CODES:
                        566 ;
                        567 ;       NONE
                        568 ;
                        569 ; SIDE EFFECTS:
                        570 ;
                        571 ;       NONE
                        572 ;
                        573 ;--
```

```
                          00CA    575 VREXAMINE::                                 ;
                 00F8 8F   BB 00CA    576         PUSHR   #^M<R3,R4,R5,R6,R7>     ; SAVE SOME REGISTERS
                    0679   30 00CE    577         BSBW    RBREAK_OUT              ; REMOVE ALL BREAKPOINTS
                           00D1    578
                           00D1    579 10$:
               53   18 A2   D0 00D1    580         MOVL    CLI$L_ADDRESS(R2),R3    ; Get address for examine
                           00D5    581 ;+
                           00D5    582 ; First, determine where specified item is: if it is an internal CPU
                           00D5    583 ; register, special handling is required; but external registers are
                           00D5    584 ; available in memory.
                           00D5    585 ;-
                           00D5    586
               1B 62   14  E1 00D5    587         BBC     #CLI$V_REG,CLI$L_FLAGS(R2),20$ ; Branch if not register
                  10   53  D1 00D9    588         CMPL    R3,#16                  ; REG ADR EXCEED 'PSL' ?
                       10  1B CODC    589         BLEQU   15$                     ; Out of range
                           00DE    590                                         ;
                           00DE    591         $DS_PRINTF_S  L^FMTINVLDREG     ; COMPLAIN ABOUT REGISTER NAME          [12]
            000000F9'EF    9F 00DE                         PUSHAB  L^FMTINVLDREG
00000000'9F    01   FB 00E4                         CALLS   #$$N, @#DS$PRINTF
                    00D0   31 00E8    592         BRW     VREXAMINEX              ; AND EXIT
                           00EE    593                                         ;
                53   6C43   DE 00EE    594 15$:    MOVAL   (AP)[R3],R3             ; Locate saved register
                     0B    11 00F2    595         BRB     30$                     ; Continue
                           00F4    596                                         ;
                07 62   1A  E0 00F4    597 20$:    BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),30$ ; Branch if not memory
           53   00000058'EF   C0 00F8    598         ADDL2   L^BASEADDRESS,R3        ; Compute memory address        [12]
                           00FF    599 30$:
                           00FF    600
                           00FF    601 ;+
                           00FF    602 ; DETERMINE SIZE OF ITEM, USE DEFAULT IF NOT SPECIFIED.
                           00FF    603 ;-
                   54   04   D0 00FF    604         MOVL    #4,R4                   ; ASSUME LONGWORD
                1F 62   0F   E0 0102    605         BBS     #CLI$V_LONG,CLI$L_FLAGS(R2),80$ ; LONGWORD SPECIFIED ?
                06 62   14   E1 0106    606         BBC     #CLI$V_REG,CLI$L_FLAGS(R2),40$  ; Skip if not register examine
                10   18 A2   D1 010A    607         CMPL    CLI$L_ADDRESS(R2),#PSLREG       ; Is the register the PSL?
                        15   13 010E    608         BEQL    80$                     ; If so, ignore data type switches
                           0110    609 40$:
                   54   02   D0 0110    610         MOVL    #2,R4                   ; ASSUME WORD
                0E 62   0E   E0 0113    611         BBS     #CLI$V_WORD,CLI$L_FLAGS(R2),80$ ; WORD SPECIFIED ?
                   54   01   D0 0117    612         MOVL    #1,R4                   ; ASSUME BYTE
                07 62   0D   E0 011A    613         BBS     #CLI$V_BYTE,CLI$L_FLAGS(R2),80$ ; BYTE SPECIFIED ?
           54   00000054'EF   D0 011E    614         MOVL    L^DFLTSIZE,R4           ; ELSE USE DEFAULT SIZE         [12]
                           0125    615 80$:
```

```
                                0125   617 ;+
                                0125   618 ; EXTRACT THE TARGET ITEM FROM MEMORY
                                0125   619 ;-
                                0125   620
          14 62    1A   E0      0125   621          BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),83$ ; If internal reg exam, do it
                   7E   D4      0129   622          CLRL    -(SP)                           ; SPACE FOR DATA
                                012B   623
                   6E   9F      012B   624          PUSHAB  (SP)                            ; WHERE TO STORE CURRENT CONTENTS
                   63   9F      012D   625          PUSHAB  (R3)                            ; AND WHERE TO FETCH IT FROM
                   54   DD      012F   626          PUSHL   R4                              ; NUMBER OF BYTES TO SHOW
    000007C3'EF    03   FB      0131   627          CALLS   #3,FETCH_STORE                  ; READ THE NECESSARY BYTES
                   55   8ED0    0138   628          POPL    R5                              ; Get data into R5
                   2F   11      013B   629          BRB     86$                             ; Continue
              54   01   D0      013D   630 83$:     MOVL    #1,R4                           ; In case NEXT used, increment by 1
              55   01   D0      0140   631          MOVL    #1,R5                           ; CMK$SGIPR code for EXAMINE function
 0F 0000FE00'EF    1C   E1      0143   632          BBC     #DSA$V_USER, L^DSA$GL_FLAGS, 85$        ; User mode?
        00000000'EF 9F   14B     633          PUSHAB  L^T_EMESSG1                     ; User mode error message               [12]
    00000000'EF    01   FB      0151   634          CALLS   #1,DS$PRINTF                    ; Type it
                   64   11      0158   635          BRB     VREXAMINEX                      ; Leave this place
                                015A   636 85$:     CMK     SGIPR                           ; Store/Get IPR value via CHMK trap
                   7E   DC      015A                MOVPSL  -(SP)
          06 6E    1A   E0      015C                BBS     #PSL$V_IS, (SP), 30000$
                   8E   D4      0160                CLRL    (SP)+
                   0A   BC      0162                CHMK    #CMK$_SGIPR
                   03   11      0164                BRB     30001$
                0824    30      0166        30000$: BSBW    CMK$SGIPR
                                0169        30001$:
              55   51   D0      0169   637          MOVL    R1,R5                           ; Get the value into R5
            1F 50   E8      016C   638 86$:     BLBS    R0,100$                         ; Type it, if no error
       18 62    1A   E1      016F   639          BBC     #CLI$V_PREG,CLI$L_FLAGS(r2),90$ ; Exit if not IPR function
 50    000001D1'EF    9E      0173   640          MOVAB   L^T_PREGREAD,R0                 ; Set up for IPR read access error      [12]
                                017A   641          $PRINTI_S       L^FMT_BADPREG,R0,R3 ; Type error                            [12]
                   53   DD      017A                PUSHL   R3
                   50   DD      017C                PUSHL   R0
       000001E4'EF    9F      017E                PUSHAB  L^FMT_BADPREG
 00000000'EF    03   FB      0184                CALLS   #$$N, DSX$PRINTI
             0030   31      018B   642 90$:     BRW     VREXAMINEX                      ; Exit routine
               36   10      018E   643 100$:    BSBB    VRTYPEXAMINE                    ; Type out value
                                0190   644
                                0190   645 ;+
                                0190   646 ; If NEXT is not zero, decrement it and loop back for next item, unless
                                0190   647 ; examining registers and PSL has just been displayed, or examining IPRs
                                0190   648 ; and P255 has just been displayed.
                                0190   649 ;-
                                0190   650
          30 A2    D5   0190   651          TSTL    CLI$L_NEXT(R2)                  ; All specified items displayed?
                29   13   0193   652          BEQL    VREXAMINEX                      ; Yes, exit
       08 62    14   E1   0195   653          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),110$ ; Skip if not registers
       10   18 A2    D1   0199   654          CMPL    CLI$L_ADDRESS(R2),#PSLREG       ; Pointing at PSL?
                1F   13   019D   655          BEQL    VREXAMINEX                      ; Yes, time to exit
                0E   11   019F   656          BRB     120$                            ; Increment register-style
          0F 62    1A   E1   01A1   657 110$:    BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),130$ ; Skip if not IPRs
 000000FF 8F   18 A2    D1   01A5   658          CMPL    CLI$L_ADDRESS(R2),#255          ; Pointing at top IPR?
                0F   13   01AD   659          BEQL    VREXAMINEX                      ; Yes, time to exit
             18 A2    D6   01AF   660 120$:    INCL    CLI$L_ADDRESS(R2)               ; Bump to next register
                04   11   01B2   661          BRB     140$                            ; Skip to loop back
          18 A2    54   C0   01B4   662 130$:    ADDL2   R4,CLI$L_ADDRESS(R2)    ; Bump to next data item
```

```
                30 A2   D7  01B8   663 140$:   DECL    CLI$L_NEXT(R2)          ; Count this item
                 FF13   31  01BB   664          BRW     10$                     ; Do it all again
                        01BE   665
                        01BE   666 ;+
                        01BE   667 ; Return
                        01BE   668 ;-
                        01BE   669
                        01BE   670 VREXAMINEX:
                  05B4  30  01BE   671          BSBW    RBREAK_IN               ; Re-insert breakpoints
                00F8 8F   BA  01C1   672          POPR    #^M<R3,R4,R5,R6,R7>     ; Restore registers
                          05  01C5   673          RSB                             ; Return
```

```
                                01C6   675 .SBTTL  VRTYPEXAMINE    Type out location EXAMINE'd
                                01C6   676
                                01C6   677 ;+
                                01C6   678 ; DETERMINE DISPLAY RADIX TO BE USED, DEFAULT IF NOT SPECIFIED.
                                01C6   679 ;  (CONSTRUCT A CONTROL STRING FOR ''FAO'')
                                01C6   680 ;-
                                01C6   681 VRTYPEXAMINE:
                    00C0 8F     BB  01C6   682          PUSHR   #^M<R6,R7>
                                01CA   683
          0000007A'EF   4C 8F   90  01CA   684          MOVB    #^A''L'',L^FMTDIRECTIVE+10          ; SET UP FOR ADDRESS CONVERSION [12]
                  2A 62   12    E0  01D2   685          BBS     #CLI$V_HEX,CLI$L_FLAGS(R2),140$ ; BRANCH IF HEX SPECIFIED
                  56   5A 8F   90  01D6   686          MOVB    #^A'Z',R6                       ; SETUP FOR DECIMAL
                  26 62   10    E0  01DA   687          BBS     #CLI$V_DEC,CLI$L_FLAGS(R2),150$ ; BRANCH IF DECIMAL SPECIFIED
                  56   4F 8F   90  01DE   688          MOVB    #^A'O',R6                       ; SETUP OCTAL
                  1E 62   11    E0  01E2   689          BBS     #CLI$V_OCT,CLI$L_FLAGS(R2),150$ ; BRANCH IF OCTAL SPECIFIED
                                01E6   690
                  56   4F 8F   90  01E6   691          MOVB    #^A'O',R6                   ; SETUP OCTAL
              08  00000000'EF   D1  01EA   692          CMPL    L^DS$GL_RADIX,#8            ; OCTAL DEFAULT ?                   [12]
                         11     13  01F1   693          BEQL    150$                        ; YES, USE IT
                  56   5A 8F   90  01F3   694          MOVB    #^A'Z',R6                   ; SETUP DECIMAL
              0A  00000000'EF   D1  01F7   695          CMPL    L^DS$GL_RADIX,#10           ; DECIMAL DEFAULT ?                 [12]
                         04     13  01FE   696          BEQL    150$                        ; YES, USE IT
                   56   58 8F   90  0200   697 140$:    MOVB    #^A''X'',R6                 ; OTHERWIZE USE HEXIDECIMAL
                                0204   698
          00000079'FF   56      90  0204   699 150$:    MOVB    R6,L^FMTDIRECTIVE+9         ; INSERT DIRECTIVE BYTE             [12]
```

```
                                 020B    701 ;+
                                 020B    702 ; CONVERT ADDRESS (REGISTER OR MEMORY) OF ITEM TO ASCII
                                 020B    703 ; AND INSERT INTO OUTPUT BUFFER, FOLLOWED BY SYNTACTICAL SEPARATORS.
                                 020B    704 ;-
                                 020B    705
          0000007B'EF    20   D0  020B    706          MOVL    #EXAMSIZE,L^EXAMQWDBUF          ; SET UP BUFFER CHAR. COUNT     [12]
 0000007F'EF    00000083'EF   DE  0212    707          MOVAL   L^EXAMBUFFER,L^EXAMQWDBUF+4     ; SET UP BUFFER POINTER         [12]
                              021D    708
          00000107'EF    63   DE  021D    709          MOVAL   (R3),L^FAOARG                   ; SET ADDRESS TO CONVERT        [12]
                57   18 A2   D0  0224    710          MOVL    CLI$L_ADDRESS(R2),R7            ; Get address for conversion    [12]
                17 62    14   E1  0228    711          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),170$ ; Branch if not external reg.  [12]
     00000079'EF    4341 8F   B0  022C    712          MOVW    #^A"AC",L^FMTDIRECTIVE+9        ; YES, SET TO COPY A STRING     [12]
00000107'EF    0000026E'EF47   D0  0235    713          MOVL    L^REGNAMLST[R7],L^FAOARG        ; STRING ADDRESS FOR FAO        [12]
                      27   11  0241    714          BRB     200$                            ;                               [12]
                23 62    1A   E1  0243    715 170$:    BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),200$ ; Branch if not internal reg. [12]
          0000007F'FF    50 8F   90  0247    716          MOVB    #^A'P',@L^EXAMQWDBUF+4          ; Move a "P" into output        [12]
                0000007F'EF    D6  024F    717          INCL    L^EXAMQWDBUF+4                  ; Update buffer avail. address  [12]
                0000007B'EF    D7  0255    718          DECL    L^EXAMQWDBUF                    ; Update buffer avail. byte count.[1
          0000007A'EF    42 8F   90  025B    719          MOVB    #^A'B',L^FMTDIRECTIVE+10        ; Set for byte edit in radix    [12]
             00000107'EF    57   D0  0263    720          MOVL    R7,L^FAOARG                     ; Point FAO to the register name.[12
                              026A    721
                              026A    722 200$:    $FAOL_G  L^FAOLST                        ; CONVERT ADDRESS TO ASCII      [12]
00000000'GF    000000F3'EF   FA  026A                       CALLG   L^FAOLST,G^SYS$FAOL
0000007B'EF    0000007D'EF   A2  0275    723          SUBW2   L^EXAMQWDBUF+2,L^EXAMQWDBUF     ; ADJUST BUFFER DESCRIPTOR      [12]
          57   0000007D'EF   3C  0280    724          MOVZWL  L^EXAMQWDBUF+2,R7               ;  (BY NUMBER INSERTED)         [12]
          0000007F'EF    57   C0  0287    725          ADDL2   R7,L^EXAMQWDBUF+4               ;  TO REMAINING BYTES           [12]
                              028E    726
0000007F'FF    093A 8F   B0  028E    727          MOVW    #^A':   ',@L^EXAMQWDBUF+4       ; STORE <:<TAB>> AFTER NUMBER   [12]
     0000007F'EF    02   C0  0297    728          ADDL2   #2,EXAMQWDBUF+4                 ; UPDATE AVAILABLE ADDRESS
     0000007B'EF    02   A2  029E    729          SUBW2   #2,EXAMQWDBUF                   ; SHORTEN AVAILABLE LENGTH
```

```
                                02A5   731 ;+
                                02A5   732 ; IF ITEM IS NUMERIC, CONVERT IT TO ASCII INTO OUTPUT BUFFER.
                                02A5   733 ;-
                                02A5   734
                000000A3'EF   94  02A5   735          CLRB    L^PSLBUFFER                ; ERASE ANY PREVIOUS MNEMONICS          [12]
                 03 62    13  E1  02AB   736          BBC     #CLI$V_ASCII,CLI$L_FLAGS(R2),199$ ; Skip branch, if ASCII      [12]
                       008C  31  02AF   737          BRW     400$                       ; Branch if not ASCII                   [12]
                                02B2   738 199$:                                       ;                                       [12]
                 12 62    1A  E0  02B2   739          BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),300$ ; Always longword, if IPR
                 57    42 8F  90  02B6   740          MOVB    #^A'B',R7                  ; SELECT BYTE OUTPUT
                    01    54  D1  02BA   741          CMPL    R4,#1                      ; SIZE = BYTE ?
                          0D  13  02BD   742          BEQL    330$                       ; YES, USE IT
                 57    57 8F  90  02BF   743          MOVB    #^A'W',R7                  ; YES, SELECT DIRECTIVE
                    02    54  D1  02C3   744          CMPL    R4,#2                      ; SIZE = WORD ?
                          04  13  02C6   745          BEQL    330$                       ; YES, USE IT
                 57    4C 8F  90  02C8   746 300$:    MOVB    #^A'L',R7                  ; Use longword form
          00000079'EF    56  90  02CC   747 330$:    MOVB    R6,L^FMTDIRECTIVE+9        ; RADIX TO DIRECTIVE STRING             [12]
          0000007A'EF    57  90  02D3   748          MOVB    R7,L^FMTDIRECTIVE+10       ; SIZE TO DIRECTIVE STRING              [12]
          00000107'EF    55  D0  02DA   749          MOVL    R5,L^FAOARG                ; DATA ITEM FOR FAO                     [12]
                                02E1   750          $FAOL_G  L^FAOLST                  ; CONVERT ITEM TO ASCII                 [12]
00000000'GF    000000F3'EF    FA  02E1                CALLG   L^FAOLST,G^SYS$FAOL
          57    0000007D'EF    3C  02EC   751          MOVZWL  L^EXAMQWDBUF+2,R7          ; GET COUNT OF BYTES INSERTED           [12]
          0000007B'EF    57  A2  02F3   752          SUBW2   R7,L^EXAMQWDBUF            ; ADJUST BUFFER DESCRIPTOR              [12]
          0000007F'EF    57  C0  02FA   753          ADDL2   R7,L^EXAMQWDBUF+4          ; UPDATE ADDRESS                        [12]
                                0301   754
                                0301   755 ;+
                                0301   756 ; AND IF ITEM IS 'PSL', CONVERT ITS CONTENTS TO A MNEMONIC STRING.
                                0301   757 ;-
                                0301   758
                 37 62    14  E1  0301   759          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),350$ ; BRANCH IF NOT REGISTER
                 10    18 A2  D1  0305   760          CMPL    CLI$L_ADDRESS(R2),#PSLREG ; REGISTER = "PSL" ?
                          31  12  0309   761          BNEQ    350$                       ; NO, SO SKIP OUT
                                030B   762          $DS_CVTREG_S  #31,R5,L^APSLMNE,L^PSLBUFFER,#PSLBUFSZ, - ; CNVRT TO MNCS [12]
                                030B   763                  MODELST,MODELST            ; INCLUDING ACCESS MODE MNEMONICS
                          00    DD  030B                   PUSHL   #0
                          00    DD  030D                   PUSHL   #0
                          00    DD  030F                   PUSHL   #0
                          00    DD  0311                   PUSHL   #0
                00000239'EF     DF  0313                   PUSHAL  MODELST
                00000239'EF     DF  0319                   PUSHAL  MODELST
                00000050 8F     DD  031F                   PUSHL   #PSLBUFSZ
                000000A3'EF     9F  0325                   PUSHAB  L^PSLBUFFER
                000001F8'EF     9F  032B                   PUSHAB  L^APSLMNE
                          55    DD  0331                   PUSHL   R5
                          1F    DD  0333                   PUSHL   #31
          00000000'9F     0B    FB  0335                   CALLS   #11, @#DS$CVTREG
                          3F    11  033C   764 350$:    BRB     410$                       ; SKIP TO DISPLAY
```

```
                           033E    766 ;+
                           033E    767 ; ASCII examine.  Print the data in !AF format (non-printing characters appear
                           033E    768 ; as ".") and follow by hex printout.
                           033E    769 ;-
                           033E    770
                           033E    771 400$:
     56    54    01    78  033E    772          ASHL    #1,R4,R6                    ; Calculate # hex digits to type
                 55    DD  0342    773          PUSHL   R5                          ; Make data available in memory
           50    6E    9E  0344    774          MOVAB   (SP),R0                     ; For convenience, point to it
           51    7E    DE  0347    775          MOVAL   -(SP),R1                    ; Allocate a longword
                           034A    776          $FAO_S  L^FMTEXAM_ASCII,(R1),-      ; Format for typeout                    [12]
                           034A    777                   L^EXAMQWDBUF,-            ;                                       [12]
                           034A    778                   R4,R0,R6,R5
                 55    DD  034A                         PUSHL   R5
                 56    DD  034C                         PUSHL   R6
                 50    DD  034E                         PUSHL   R0
                 54    DD  0350                         PUSHL   R4
      0000007B'EF    7F  0352                         PUSHAQ  L^EXAMQWDBUF
                 61    3F  0358                         PUSHAW  (R1)
      00000118'EF    7F  035A                         PUSHAQ  L^FMTEXAM_ASCII
00000000'GF    07    FB  0360                         CALLS   #$$T2,G^SYS$FAO
                 51  8ED0  0367    779          POPL    R1                          ; Get returned length
           51    51    3C  036A    780          MOVZWL  R1,R1                       ; Clear upper word
                 8E    D4  036D    781          CLRL    (SP)+                       ; De-allocate temporary
0000007B'EF    51    A2  036F    782          SUBW2   R1,L^EXAMQWDBUF             ; Cut buffer size                       [12]
0000007F'EF    51    A0  0376    783          ADDW2   R1,L^EXAMQWDBUF+4           ; Increase buffer address               [12]
                           037D    784
                           037D    785 ;+
                           037D    786 ; THEN PRINT THE OUTPUT BUFFER(S).
                           037D    787 ;-
                           037D    788
0000007B'EF    20  0000007B'EF    A3  037D    789 410$:    SUBW3   L^EXAMQWDBUF,#EXAMSIZE,L^EXAMQWDBUF ; SET UP STRING LENGTH  [12]
     0000007F'EF  00000083'EF    DE  0389    790          MOVAL   L^EXAMBUFFER,L^EXAMQWDBUF+4 ; SET UP BUFFER POINTER      [12]
                           0394    791          $DS_PRINTF_S  L^FMTEXAM, -          ;                                       [12]
                           0394    792                   #EXAMQWDBUF, -            ; Address and contents
                           0394    793                   #PSLBUFFER                ; Register Mnemonics
                 000000A3'8F    DD  0394                         PUSHL   #PSLBUFFER
                 0000007B'8F    DD  039A                         PUSHL   #EXAMQWDBUF
                 0000012B'EF    9F  03A0                         PUSHAB  L^FMTEXAM
           00000000'9F    03    FB  03A6                         CALLS   #$$N, @#DS$PRINTF
                 00C0  8F    BA  03AD    794          POPR    #^M<R6,R7>          ; Restore registers
                             05  03B1    795          RSB                         ; Return
```

```
                        798          .SBTTL  DEPOSIT MEMORY AND REGISTERS SUBROUTINE
                        799 ;++
                        800 ; FUNCTIONAL DESCRIPTION:
                        801 ;
                        802 ;       LOCATES AND MODIFIES MEMORY OR GENERAL REGISTERS AS SPECIFIED
                        803 ;       BY THE OPERATOR.
                        804 ;       ALL SUPERVISOR OVERHEAD (I.E., STACK & REGISTER USAGE)
                        805 ;       IS TRANSPARENT TO THE OPERATOR.
                        806 ;
                        807 ; CALLING SEQUENCE:
                        808 ;
                        809 ;       BSBW    VRDEPOSIT
                        810 ;
                        811 ; INPUT PARAMETERS:
                        812 ;
                        813 ;       R2      = BASE OF CLI DATA FRAME.
                        814 ;       CLI$L_FLAGS(R2)         ; SIZE SPECIFIERS
                        815 ;       CLI$L_ADDRESS(R2)       ; ADDRESS SPECIFIER
                        816 ;       CLI$L_DATA(R2)          ; ITEM TO BE STORED
                        817 ;
                        818 ; IMPLICIT INPUTS:
                        819 ;
                        820 ;       (AP) = BASE ADDRESS OF PROGRAM REGISTER CONTEXT
                        821 ;       DFLTSIZE                        ; DEFAULT SIZE IN BYTES
                        822 ;
                        823 ; OUTPUT PARAMETERS:
                        824 ;
                        825 ;       NONE
                        826 ;
                        827 ; IMPLICIT OUTPUTS:
                        828 ;
                        829 ;       (AP) = BASE ADDRESS OF PROGRAM REGISTER CONTEXT (MODIFIED)
                        830 ;
                        831 ; COMPLETION CODES:
                        832 ;
                        833 ;       NONE
                        834 ;
                        835 ; SIDE EFFECTS:
                        836 ;
                        837 ;       NONE
                        838 ;
                        839 ;--
```

```
                                03B2    841 VRDEPOSIT::                                 ;
                          18    BB  03B2    842             PUSHR   #^M<R3,R4>              ; SAVE SOME REGISTERS
                                03B4    843
                                03B4    844 10$:
                    53    18 A2   D0  03B4    845             MOVL    CLI$L_ADDRESS(R2),R3    ; Get address
                                03B8    846
                                03B8    847 ;+
                                03B8    848 ; FIRST, DETERMINE LOCATION OF ITEM SPECIFIED.  IF A REGISTER,
                                03B8    849 ; IT'S STORED SOMEWHERE IN MEMORY AT THIS POINT IN TIME.
                                03B8    850 ;-
                                03B8    851
                    1B 62    14 E1  03B8    852             BBC     #CLI$V_REG,CLI$L_FLAGS(R2),20$ ; Branch if not external reg
                       10    53 D1  03BC    853             CMPL    R3,#16                  ; REG ADR EXCEED 'PSL' ?
                             10 1B  03BF    854             BLEQU   15$
                                03C1    855
                                03C1    856             $DS_PRINTF_S  L^FMTINVLDREG     ; COMPLAIN ABOUT REGISTER NAME          [12]
                 000000F9'EF    9F  03C1                            PUSHAB  L^FMTINVLDREG
            00000000'9F    01    FB  03C7                            CALLS   #$$N, @#DS$PRINTF
                          00D9  31  03CE    857             BRW     VRDEPOSITX              ; AND EXIT
                                03D1    858                                             ;
                    53    6C43  DE  03D1    859 15$:        MOVAL   (AP)[R3],R3             ; Calculate real address
                             0B 11  03D5    860             BRB     30$                     ; Continue
                                03D7    861
                    07 62    1A E0  03D7    862 20$:        BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),30$ ; Branch if not internal reg.
              53    00000058'EF    C0  03DB    863             ADDL2   L^BASEADDRESS,R3        ; Add base offset if memory             [12]
                                03E2    864 30$:
                                03E2    865
                                03E2    866 ;+
                                03E2    867 ; DETERMINE SIZE OF ITEM, USE DEFAULT IF NOT SPECIFIED.
                                03E2    868 ;-
                                03E2    869
                       54    04  D0  03E2    870             MOVL    #4,R4                   ; YES, SET SIZE IN BYTES
                    15 62    0F  E0  03E5    871             BBS     #CLI$V_LONG,CLI$L_FLAGS(R2),80$ ; BRANCH IF LONGWORD SPECIFIED
                       54    02  D0  03E9    872             MOVL    #2,R4                   ; YES, SET SIZE IN BYTES
                    0E 62    0E  E0  03EC    873             BBS     #CLI$V_WORD,CLI$L_FLAGS(R2),80$ ; BRANCH IF WORD SPECIFIED
                       54    01  D0  03F0    874             MOVL    #1,R4                   ; YES, SET SIZE IN BYTES
                    07 62    0D  E0  03F3    875             BBS     #CLI$V_BYTE,CLI$L_FLAGS(R2),80$ ; BRANCH IF BYTE SPECIFIED
              54    00000054'EF    D0  03F7    876             MOVL    L^DFLTSIZE,R4           ; ELSE USE DEFAULT SIZE IN BYTES        [12]
                                03FE    877 80$:
```

```
                                03FE    879 ;+
                                03FE    880 ; INSERT THE TARGET ITEM INTO MEMORY
                                03FE    881 ;-
                                03FE    882
                     0349   30  03FE    883          BSBW    RBREAK_OUT                      ; REMOVE ALL BREAKPOINTS
                                0401    884
                10 62   1A   E0  0401    885          BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),90$ ; If internal CPU reg, do it
                       63   9F  0405    886          PUSHAB  (R3)                            ; ADDRESS TO STORE IT
                     1C A2  9F  0407    887          PUSHAB  CLI$L_DATA(R2)                  ; ADDRESS OF DATA TO STORE
                       54   DD  040A    888          PUSHL   R4                              ; NUMBER OF BYTES TO STORE
          000007C3'EF   03   FB  040C    889          CALLS   #3,FETCH_STORE                  ; STORE THOSE BYTES THERE
                       33   11  0413    890          BRB     100$                            ; Continue
                  54   01   D0  0415    891 90$:     MOVL    #1,R4                           ; In case NEXT option, increment by 1
                       55   D4  0418    892          CLRL    R5                              ; CMK$SGIPR code for DEPOSIT function
                51   1C A2   D0  041A    893          MOVL    CLI$L_DATA(R2),R1               ; Data value to store
          10 0000FE00'EF   1C   E1  041E    894          BBC     #DSA$V_USER, L^DSA$GL_FLAGS, 95$        ; User mode?
                00000000'EF   9F  0426    895          PUSHAB  L^T_EMESSG1                     ; User mode error message               [12]
             00000000'EF   01   FB  042C    896          CALLS   #1,DS$PRINTF                    ; Type it
                     0074   31  0433    897          BRW     VRDEPOSITX                      ; Leave this place
                                0436    898 95$:     CMK     SGIPR                           ; Store/Get IPR value via CHMK trap
                       7E   DC  0436                   MOVPSL  -(SP)
                06 6E   1A   E0  0438                   BBS     #PSL$V_IS, (SP), 30002$
                       8E   D4  043C                   CLRL    (SP)+
                       0A   BC  043E                   CHMK    #CMK$_SGIPR
                       03   11  0440                   BRB     30003$
                     0548   30  0442         30002$:   BSBW    CMK$SGIPR
                                0445         30003$:
                  55   51   D0  0445    899          MOVL    R1,R5                           ; Save read-back verify value
                                0448    900 100$:
                                0448    901
                       01   BB  0448    902          PUSHR   #^M<R0>                         ; SAVE RETURN CODE
                     0328   30  044A    903          BSBW    RBREAK_IN                       ; PUT BREAKPOINTS BACK
                       01   BA  044D    904          POPR    #^M<R0>                         ; RESTORE RETURN CODE
                                044F    905
                2B 50   E8   E8  044F    906          BLBS    R0,120$                         ; Continue if no error
             54 62   1A   E1   E1  0452    907          BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),VRDEPOSITX ; Exit if not IPR
             09 50   01   E0   E0  0456    908          BBS     #1,R0,T10$                      ; Was it the DEPOSIT that failed?
     50   000001DB'EF   9E   9E  045A    909          MOVAB   L^T_PREGWRITE,R0                ; Yes--set up correct error message     [12]
                       07   11  0461    910          BRB     115$                            ; Continue                              [12]
     50   000001D1'EF   9E   9E  0463    911 110$:    MOVAB   L^T_PREGREAD,R0                 ; Error was on read-back                [12]
                                046A    912 115$:    $PRINTI_S       L^FMT_BADPREG,R0,R3 ; Type error message                [12]
                       53   DD  046A                   PUSHL   R3
                       50   DD  046C                   PUSHL   R0
             000001E4'EF   9F  046E                   PUSHAB  L^FMT_BADPREG
          00000000'EF   03   FB  0474                   CALLS   #$$N, DSX$PRINTI
                       2D   11  047B    913          BRB     VRDEPOSITX                      ; Exit routine now.
                                047D    914
                                047D    915 ;+
                                047D    916 ; IF "NEXT" IS NOT ZERO, DECREMENT IT AND LOOP BACK FOR NEXT ITEM.
                                047D    917 ;  (EXCEPT WHEN DEPOSITING REGISTERS AND "PSL" HAS JUST BEEN MODIFIED)
                                047D    918 ;-
                                047D    919
                30 A2   D5  047D    920 120$:    TSTL    CLI$L_NEXT(R2)                  ; All specified items displayed?
                     1F   13  0480    921          BEQL    150$                            ; YES, EXIT
                                0482    922                                             ;
             08 62   14   E1  0482    923          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),125$ ; Skip if not registers
             10   18 A2   D1  0486    924          CMPL    CLI$L_ADDRESS(R2),#PSLREG       ; POINTING AT "PSL" ?
```

```
                      15   13  048A   925          BEQL    150$                            ; YES, THEN IT'S TIME TO EXIT
                      04   11  048C   926          BRB     128$                            ; Increment
               05 62  1A   E1  048E   927 125$:     BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),130$ ; Skip if not IPRs
                  18  A2   D6  0492   928 128$:     INCL    CLI$L_ADDRESS(R2)               ; Bump to next register
                      04   11  0495   929          BRB     140$                            ; AND SKIP TO LOOP BACK
                                0497   930                                                  ;
               18 A2  54   C0  0497   931 130$:     ADDL2   R4,CLI$L_ADDRESS(R2)            ; BUMP TO NEXT DATA ITEM
                  30  A2   D7  049B   932 140$:     DECL    CLI$L_NEXT(R2)                  ; DO ACCOUNTING
                  FF13    31  049E   933          BRW     10$                             ; GO DO WHOLE THING AGAIN
                                04A1   934
               02 62  1A   E0  04A1   935 150$:     BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),160$ ; Branch if IPR
                      03   11  04A5   936          BRB     VRDEPOSITX                      ; Then exit
                    FD1C   30  04A7   937 160$:     BSBW    VRTYPEXAMINE                    ; Only do display part of VREXAMINE
                                04AA   938
                                C4AA   939 VRDEPOSITX:                                   ;
                      18   BA  04AA   940          POPR    #^M<R3,R4>                      ; RESTORE REGISTERS
                           05  04AC   941          RSB                                     ; RETURN
```

```
                        04AD   943          .SBTTL  SET BREAKPOINT SUBROUTINE
                        04AD   944 ;++
                        04AD   945 ; FUNCTIONAL DESCRIPTION:
                        04AD   946 ;
                        04AD   947 ;       SAVES AN 'OPCODE' FROM THE SPECIFIED ADDRESS AND REPLACES
                        04AD   948 ;       IT WITH A BREAKPOINT CODE
                        04AD   949 ;
                        04AD   950 ; CALLING SEQUENCE:
                        04AD   951 ;
                        04AD   952 ;       BSBW    VRSETBRK
                        04AD   953 ;
                        04AD   954 ; INPUT PARAMETERS:
                        04AD   955 ;
                        04AD   956 ;       CLI$L_ADDRESS(DS$GL_CLIBASE)
                        04AD   957 ;       CLI$L_FLAGS(DS$GL_CLIBASE)
                        04AD   958 ;
                        04AD   959 ; IMPLICIT INPUTS:      NONE
                        04AD   960 ;
                        04AD   961 ; OUTPUT PARAMETERS:    NONE
                        04AD   962 ;
                        04AD   963 ; IMPLICIT OUTPUTS:     NONE
                        04AD   964 ;
                        04AD   965 ; COMPLETION CODES:     NONE
                        04AD   966 ;
                        04AD   967 ; SIDE EFFECTS:         NONE
                        04AD   968 ;--
```

```
                                04AD    970 VRSETBRK::
                          07    BB  04AD    971         PUSHR   #^M<R0,R1,R2>              ; SAVE WORKING REGISTERS
                        0298    30  04AF    972         BSBW    RBREAK_OUT                ; REMOVE ALL BREAKPOINTS
          52  00000000'EF      DE  04B2    973         MOVAL   L^DS$GL_CLIBASE, R2       ; GET CLI ARGUMENT FRAME BASE.      [12]
                      62    17  E4  04B9    974         BBSC    #CLI$V_KERNEL,CLI$L_FLAGS(R2) -
                          27        04BC    975                 ,10$                      ; SET HIDDEN BRK IF KERNEL REQ
                                    04BD    976
                00000058'EF    C1  04BD    977         ADDL3   L^BASEADDRESS,-           ;                                   [12]
                 50    18 A2      04C3    978                 CLI$L_ADDRESS(R2), R0     ; ADDRESS OF DESIRED BREAKPOINT
                        00FE    30  04C6    979         BSBW    FIND_BPT                  ; SEE IF IT'S ALREADY THERE
                       31 50    E8  04C9    980         BLBS    R0,VRSETBRKX              ; FOUND, EXIT QUIETLY
                                    04CC    981
                          50    D4  04CC    982         CLRL    R0                        ; TRY FOR AN EMPTY SLOT
                        00F6    30  04CE    983         BSBW    FIND_BPT                  ; LOCATE EMPTY SLOT
                       1A 50    E9  04D1    984         BLBC    R0, 20$                   ; BRANCH IF NO FREE SLOTS
          18 A2  00000058'EF    C1  04D4    985         ADDL3   L^BASEADDRESS, -          ;                                   [12]
                00000014'EF41       04DC
                                    04E2    986                 CLI$L_ADDRESS(R2), -      ;                                   [12]
                                    04E2    987                 L^DS$AA_BPTADDR[R1]       ; STORE ADDRESS OF BPT IN TABLE     [12]
                          19    11  04E2    988         BRB     VRSETBRKX                 ; EXIT
                                    04E4    989
00000014'EF     18 A2       D0  04E4    990 10$:    MOVL    CLI$L_ADDRESS(R2), -
                                    04EC    991                 L^DS$AA_BPTADDR           ; SET HIDDEN BREAKPOINT             [12]
                          0F    11  04EC    992         BRB     VRSETBRKX                 ; EXIT
                                    04EE    993 20$:
                                    04EE    994         $DS_PRINTF_S  L^FMTNOBPTSLEFT,#BPTMAX ; NO MORE BREAK POINTS          [12]
                          0F    DD  04EE                        PUSHL   #BPTMAX
                00000139'EF     9F  04F0                        PUSHAB  L^FMTNOBPTSLEFT
             00000000'9F      02  FB  04F6                        CALLS   #$$N, @#DS$PRINTF
                                    04FD    995
                                    04FD    996 VRSETBRKX:
                        0275    30  04FD    997         BSBW    RBREAK_IN                 ; PUT ALL BREAKPOINTS BACK IN
                          07    BA  0500    998         POPR    #^M<R0,R1,R2>             ; RESTORE REGISTERS
                                05  0502    999         RSB                               ; RETURN
```

```
0503  1001          .SBTTL  CLEAR BREAKPOINT SUBROUTINE
0503  1002 ;++
0503  1003 ; FUNCTIONAL DESCRIPTION:
0503  1004 ;
0503  1005 ;       RESTORES THE SAVED 'OPCODE' TO THE SPECIFIED ADDRESS,
0503  1006 ;       IF IT IS A KNOWN BREAKPOINT
0503  1007 ;
0503  1008 ; CALLING SEQUENCE:
0503  1009 ;
0503  1010 ;       BSBW    VRCLRBRK
0503  1011 ;
0503  1012 ; INPUT PARAMETERS:
0503  1013 ;
0503  1014 ;       R2      = BASE OF CLI DATA FRAME.
0503  1015 ;       CLI$L_ADDRESS(R2)
0503  1016 ; IMPLICIT INPUTS:
0503  1017 ;
0503  1018 ;       CLI$L_DATA(R2)          ; IF -1, INDICATES "ALL"
0503  1019 ;
0503  1020 ; OUTPUT PARAMETERS:    NONE
0503  1021 ;
0503  1022 ; IMPLICIT OUTPUTS:     NONE
0503  1023 ;
0503  1024 ; COMPLETION CODES:     NONE
0503  1025 ;
0503  1026 ; SIDE EFFECTS:         NONE
0503  1027 ;--
```

```
                          0503  1029 VRCLRBRK::
                    07    BB  0503  1030          PUSHR   #^M<R0,R1,R2>            ; SAVE WORKING REGISTERS
                  0242    30  0505  1031          BSBW    RBREAK_OUT              ; REMOVE ALL BREAKPOINTS
          52  00000000'EF   DE  0508  1032          MOVAL   L^DS$GL_CLIBASE, R2     ; BASE CLI DATA BASE                    [12]
                          050F  1033
      FFFFFFFF 8F   1C A2   D1  050F  1034          CMPL    CLI$L_DATA(R2),#-1      ; CHECK FOR 'ALL' ARGUMENT
                    27    13  0517  1035          BEQL    30$                     ; YES, GO CLEAR THEM ALL
                          0519  1036
   50   18 A2   00000058'EF   C1  0519  1037          ADDL3   L^BASEADDRESS, -        ;                                       [12]
                          0522  1038                  CLI$L_ADDRESS(R2),R0    ; GET SPECIFIED ADDRESS
                  00A2    30  0522  1039          BSBW    FIND_BPT                ; LOCATE IT
                09 50     E9  0525  1040          BLBC    R0,10$                  ; NOT FOUND!
          00000014'EF41   D4  0528  1041          CLRL    L^DS$AA_BPTADDR[R1]     ; REMOVE BREAKPOINT                     [12]
                    1C    11  052F  1042          BRB     VRCLRBRKX               ; AND EXIT
                          0531  1043 10$:
                          0531  1044          $DS_PRINTF_S  L^FMTBPTNOTSET    ; WAS NOT SET !                         [12]
             000000DE'EF   9F  0531                      PUSHAB  L^FMTBPTNOTSET
          00000000'9F   01  FB  0537                     CALLS   #$$N, @#DS$PRINTF
                    0D    11  053E  1045          BRB     VRCLRBRKX               ; EXIT
                          0540  1046
                          0540  1047 30$:
                51    0F  D0  0540  1048          MOVL    #BPTMAX,R1              ; MAX NUMBER OF BPT'S
          00000014'EF41   D4  0543  1049 40$:     CLRL    L^DS$AA_BPTADDR[R1]     ; FREE UP THIS BREAKPOINT               [12]
                  F6 51   F5  054A  1050          SOBGTR  R1,40$                  ; CHECK WHOLE TABLE (EXCEPT [0])
                          054D  1051
                          054D  1052 VRCLRBRKX:                                   ;
                  0225    30  054D  1053          BSBW    RBREAK_IN               ; PUT BREAKPOINTS BACK IN
                    07    BA  0550  1054          POPR    #^M<R0,R1,R2>           ; RESTORE REGISTERS
                          05  0552  1055          RSB                             ; RETURN
```

```
                                0553  1057          .SBTTL  DSP$REMOVEBREAK Remove breakpoints
                                0553  1058 ;++
                                0553  1059 ;
                                0553  1060 ; FUNCTIONAL DESCRIPTION:
                                0553  1061 ;
                                0553  1062 ;       This routine can be called to remove breakpoints from
                                0553  1063 ;       a section of memory.
                                0553  1064 ;
                                0553  1065 ; CALLING SEQUENCE:
                                0553  1066 ;
                                0553  1067 ;       CALLS/G arglist,DSP$REMOVEBREAK
                                0553  1068 ;
                                0553  1069 ; INPUT PARAMETERS:
                                0553  1070 ;
                                0553  1071 ;       0       #2
                                0553  1072 ;       4       Lowest address of range
                                0553  1073 ;       8(AP)   Highest address of range
                                0553  1074 ;
                                0553  1075 ; IMPLICIT INPUTS:
                                0553  1076 ;
                                0553  1077 ;       DSA$AA_BPTADDR[n]
                                0553  1078 ;
                                0553  1079 ; OUTPUT PARAMETERS:    NONE
                                0553  1080 ;
                                0553  1081 ; IMPLICIT OUTPUTS:
                                0553  1082 ;
                                0553  1083 ;       Some breakpoints are removed from DSA$AA_BPTADDR.
                                0553  1084 ;
                                0553  1085 ; COMPLETION CODES:     NONE
                                0553  1086 ;
                                0553  1087 ; SIDE EFFECTS:        NONE
                                0553  1088 ;--
                                0553  1089
                        000C    0553  1090 .ENTRY  DSP$REMOVEBREAK,^M<R2,R3>
                                0555  1091
              01F2      30      0555  1092          BSBW    RBREAK_OUT              ; Put opcodes back in memory
53  00000014'EF        9E      0558  1093          MOVAB   L^DS$AA_BPTADDR,R3      ; Base BPT address table                [12]
      51    04 AC       7D      055F  1094          MOVQ    4(AP),R1                ; Get low and high limits
         50   0F       D0      0563  1095          MOVL    #BPTMAX,R0              ; Get length of breakpoint table
                                0566  1096
       6340   51       D1      0566  1097 10$:     CMPL    R1,(R3)[R0]             ; Compare lower with this BPT addr
              09       1A      056A  1098          BGTRU   20$                     ; Branch if lower GTR this BPT
       6340   52       D1      056C  1099          CMPL    R2,(R3)[R0]             ; Compare higher with this BPT addr
              03       1B      0570  1100          BLEQU   20$                     ; Branch if higher LSS this BPT
            6340       D4      0572  1101          CLRL    (R3)[R0]                ; Remove the BPT
          EE 50        F5      0575  1102 20$:     SOBGTR  R0,10$                  ; Count and continue
                                0578  1103
            01FA       30      0578  1104          BSBW    RBREAK_IN               ; Put remaining breakpoints back
                       04      057B  1105          RET                             ; Return
```

```
                                057C  1107          .SBTTL  SHOW BREAKPOINTS SUBROUTINE
                                057C  1108 ;++
                                057C  1109 ; FUNCTIONAL DESCRIPTION:
                                057C  1110 ;
                                057C  1111 ;       DISPLAYS ALL CURRENT BREAKPOINTS TO THE OPERATOR
                                057C  1112 ;
                                057C  1113 ; CALLING SEQUENCE:
                                057C  1114 ;
                                057C  1115 ;       BSBW    VRSHOWBRK
                                057C  1116 ;
                                057C  1117 ; INPUT PARAMETERS:     NONE
                                057C  1118 ;
                                057C  1119 ; IMPLICIT INPUTS:
                                057C  1120 ;
                                057C  1121 ;       DS$AA_BPTADDR                   ; ARRAY OF KNOWN BREAKPOINTS
                                057C  1122 ;
                                057C  1123 ; OUTPUT PARAMETERS:    NONE
                                057C  1124 ;
                                057C  1125 ; IMPLICIT OUTPUTS:     NONE
                                057C  1126 ;
                                057C  1127 ; COMPLETION CODES:     NONE
                                057C  1128 ;
                                057C  1129 ; SIDE EFFECTS:         NONE
                                057C  1130 ;--
                                057C  1131
                                057C  1132 VRSHOWBRK::                          ;
            0C   BB             057C  1133          PUSHR   #^M<R2,R3>          ; NEED A TEMPORARY
            53   D4             057E  1134          CLRL    R3                  ; SO FAR NO BREAK POINTS SET
                                0580  1135
          52    0F   D0         0580  1136          MOVL    #BPTMAX,R2          ; SET UP TABLE LENGTH
 00000014'EF42   D5             0583  1137 10$:     TSTL    L^DS$AA_BPTADDR[R2] ; SCAN BREAKPOINT ADDRESSES             [12]
            25   13             058A  1138          BEQL    30$                 ; SKIP UNTIL FOUND A VALID BPT
                                058C  1139
       0D 53    00   E2         058C  1140          BBSS    #0,R3,20$           ; IF HEADER ALREADY OUT, SKIP
                                0590  1141          $DS_PRINTF_S  L^FMTBPTLST   ; DISPLAY A SMALL HEADER            [12]
     000000B2'EF    9F          0590                        PUSHAB  L^FMTBPTLST
00000000'9F    01   FB          0596                        CALLS   #$$N, @#DS$PRINTF
                                059D  1142 20$:
                                059D  1143          $DS_PRINTF_S  L^FMTBPTADR,L^DS$AA_BPTADDR[R2] ; DISPLAY 'PC' OF BRKPNT  [12]
 00000014'EF42   DD             059D                        PUSHL   L^DS$AA_BPTADDR[R2]
    000000A7'EF    9F          05A4                        PUSHAB  L^FMTBPTADR
00000000'9F    02   FB          05AA                        CALLS   #$$N, @#DS$PRINTF
                                05B1  1144 30$:
          CF 52    F5          05B1  1145          SOBGTR  R2,10$              ; GO TRY SOME MORE (EXCEPT [0])
          0D 53    E8          05B4  1146          BLBS    R3,VRSHOWBRKX       ; EXIT IF THERE WERE ANY BREAKPOINTS
                                05B7  1147
                                05B7  1148          $DS_PRINTF_S    L^FMTBPTNONE    ; 'NONE CURRENTLY SET'          [12]
     000000C9'EF    9F          05B7                        PUSHAB  L^FMTBPTNONE
00000000'9F    01   FB          05BD                        CALLS   #$$N, @#DS$PRINTF
                                05C4  1149
                                05C4  1150 VRSHOWBRKX:
            0C   BA             05C4  1151          POPR    #^M<R2,R3>          ; RESTORE TEMPORARIES
                 05             05C6  1152          RSB                         ; RETURN
```

```
                                05C7  1154          .SBTTL  FIND_BPT        FIND BPT IN BPT TABLE
                                05C7  1155 ;++
                                05C7  1156 ; FUNCTIONAL DESCRIPTION:
                                05C7  1157 ;
                                05C7  1158 ;       Locate a specific breakpoint in the breakpoint table.
                                05C7  1159 ;
                                05C7  1160 ; CALLING SEQUENCE:
                                05C7  1161 ;
                                05C7  1162 ;       BSBW    FIND_BPT
                                05C7  1163 ;
                                05C7  1164 ; INPUT PARAMETERS:
                                05C7  1165 ;
                                05C7  1166 ;       R0      Address of desired BPT or zero if empty slot wanted
                                05C7  1167 ;
                                05C7  1168 ; IMPLICIT INPUTS:      NONE
                                05C7  1169 ;
                                05C7  1170 ; OUTPUT PARAMETERS:
                                05C7  1171 ;
                                05C7  1172 ;       R1      Index of BPT entry  (DS$AA_BPTADDR[x] as a LONGWORD)
                                05C7  1173 ;
                                05C7  1174 ; IMPLICIT OUTPUTS:     NONE
                                05C7  1175 ;
                                05C7  1176 ; RETURN CODES:
                                05C7  1177 ;
                                05C7  1178 ;       R0      SS$_NORMAL IF FOUND ELSE ZERO
                                05C7  1179 ;
                                05C7  1180 ; SIDE EFFECTS:         NONE
                                05C7  1181 ;--
                                05C7  1182
                                05C7  1183 FIND_BPT:
            51    0F    D0      05C7  1184          MOVL    #BPTMAX,R1              ; START AT THE TOP
                                05CA  1185 10$:
00000014'EF41     50    D1      05CA  1186          CMPL    R0,L^DS$AA_BPTADDR[R1]  ; IS IT?                                [12]
                  06    13      05D2  1187          BEQL    20$                     ; YES, FOUND IT
               F3 51    F5      05D4  1188          SOBGTR  R1,10$                  ; CONTINUE UNTIL AFTER CHECKED [1]
                  50    D4      05D7  1189          CLRL    R0                      ; SET NOT FOUND
                        05      05D9  1190          RSB                             ; RETURN NOT FOUND
            50    01    D0      05DA  1191 20$:     MOVL    #SS$_NORMAL,R0          ; SUCCESS
                        05      05DD  1192          RSB                             ; RETURN
```

```
                                05DE  1194          .SBTTL  DEBUGGER CONDITION HANDLER
                                05DE  1195 ;++
                                05DE  1196 ; FUNCTIONAL DESCRIPTION:
                                05DE  1197 ;
                                05DE  1198 ;       This routine is the secondary condition handler.
                                05DE  1199 ;       It is here to allow the supervisor to capture BPT and TBIT
                                05DE  1200 ;       traps. It discards all other conditions.
                                05DE  1201 ;
                                05DE  1202 ; CALLING SEQUENCE:
                                05DE  1203 ;
                                05DE  1204 ;       STANDARD PROCEEDURE CALL WITH STANDARD EXCEPTION ARGS
                                05DE  1205 ;
                                05DE  1206 ; INPUT PARAMETERS:
                                05DE  1207 ;
                                05DE  1208 ;       STANDARD EXCEPTION SIGNAL AND MECHANISM ARRAYS
                                05DE  1209 ;
                                05DE  1210 ;
                                05DE  1211 ; IMPLICIT INPUTS:      NONE
                                05DE  1212 ;
                                05DE  1213 ; OUTPUT PARAMETERS:    NONE
                                05DE  1214 ;
                                05DE  1215 ; IMPLICIT OUTPUTS:     NONE
                                05DE  1216 ;
                                05DE  1217 ; RETURN CODES:
                                05DE  1218 ;
                                05DE  1219 ;       SS$_CONTINUE    if one of our break points or T-bit.
                                05DE  1220 ;       SS$_RESIGNAL    otherwise
                                05DE  1221 ;
                                05DE  1222 ; SIDE EFFECTS:
                                05DE  1223 ;
                                05DE  1224 ;       UNKNOWN SINCE THE OPERATOR CAN CHANGE THINGS WHILE IN COMMAND MODE
                                05DE  1225 ;--
                          0000  05DE  1226 .ENTRY  COND_DEBUG,^M<>
                                05E0  1227
          50    0000'8F     3C  05E0  1228          MOVZWL  #SS$_RESIGNAL,R0        ; ASSUME IT'S SOMETHING ELSE
            51    04 BC     DE  05E5  1229          MOVAL   @4(AP), R1              ; GET SIGNAL ARRAY POINTER
04 A1    00000000'8F     D1  05E9  1230          CMPL    #SS$_BREAK, 4(R1)       ; IS THIS A BREAKPOINT CONDITION?
                  04     12  05F1  1231          BNEQ    20$                     ; NO, TRY T-BIT
                0010     30  05F3  1232          BSBW    DBG_BREAK               ; HANDLE IT
                  04     05F6  1233          RET                             ; RETURN
                         05F7  1234 20$:
04 A1    0000v000'8F     D1  05F7  1235          CMPL    #SS$_TBIT, 4(R1)        ; IS THIS A T-BIT TRAP?
                  03     12  05FF  1236          BNEQ    30$                     ; NO, EXIT
                00D2     30  0601  1237          BSBW    DBG_TBIT                ; HANDLE T-BIT
                         0604  1238 30$:
                  04     0604  1239          RET                             ; RETURN
```

```
                     0605  1241          .SBTTL  BREAKPOINT CONDITION SUBROUTINE
                     0605  1242 ;++
                     0605  1243 ; FUNCTIONAL DESCRIPTION:
                     0605  1244 ;
                     0605  1245 ;       This routine setups the context necessary and announces the break point.
                     0605  1246 ;
                     0605  1247 ; CALLING SEQUENCE:
                     0605  1248 ;
                     0605  1249 ;       BSBW    DBG_BREAK
                     0605  1250 ;
                     0605  1251 ; INPUT PARAMETERS:
                     0605  1252 ;
                     0605  1253 ;       STANDARD EXCEPTION ARGS
                     0605  1254 ;
                     0605  1255 ; IMPLICIT INPUTS:      NONE
                     0605  1256 ;
                     0605  1257 ; OUTPUT PARAMETERS:    NONE
                     0605  1258 ;
                     0605  1259 ; IMPLICIT OUTPUTS:     NONE
                     0605  1260 ;
                     0605  1261 ; COMPLETION CODES:     NONE
                     0605  1262 ;
                     0605  1263 ; SIDE EFFECTS:         NONE
                     0605  1264 ;--
```

```
M 14  Main code
N 14  Main code
B 15  Main code
C 15  Main code
D 15  Main code
E 15  Main code
F 15  Main code
G 15  Master routine
H 15  Master routine
I 15  *** DISPAT Diagnostic test seq
J 15  *** DISPAT Diagnostic test seq
K 15  *** DISPAT Diagnostic test seq
L 15  *** DISPAT Diagnostic test seq
M 15  *** DISPAT Diagnostic test seq
N 15  Libraries and Symbol Definitio
B 16  Work Declarations
C 16  Data Declarations
D 16  Data Declarations
E 16  Data Declarations
F 16  Dispatch Routine
G 16  Dispatch Routine
H 16  Dispatch Routine
I 16  Dispatch Routine
J 16  Dispatch Routine
K 16  Dispatch Routine
L 16  Dispatch Routine
```

```
                04 AE   40000010 8F    CA   068D  1320          BICL2   #PSL$M_TP!PSL$M_TBIT,4(SP) ; MAY HAVE BEEN SET BY OPERATOR      [12]
                   00000068'EF   8E    7D   0695  1321          MOVQ    (SP)+,L^NEW_PC_PSL          ; PC/PSL TO A SAFE PLACE             [12]
                            0C A0    DC   069C  1322          MOVPSL  12(R0)                      ; PUT CURR PSL ON ORIGINAL STACK
                      08 A0   A8'AF  9E   069F  1323          MOVAB   B^200$,8(R0)                ; PUT LOCAL PC ON ORIGINAL STACK
                                       06A4  1324
                           50   00'   D0   06A4  1325          MOVL    S^#SS$_CONTINUE,R0          ; INDICATE "HANDLED"
                                      05   06A7  1326          RSB                                 ; RETURN TO EXCEPTION HANDLER
                                       06A8  1327
                                       06A8  1328 ;+
                                       06A8  1329 ; THE FOLLOWING CODE IS EXECUTED BY THE 'REI' AT THE END OF THE
                                       06A8  1330 ; EXCEPTION HANDLER.  THIS FACILITATES THE MODIFICATION OF THE
                                       06A8  1331 ; 'SP', 'PC' & 'PSL' (BY THE OPERATOR) BEFORE CONTROL RETURNS TO
                                       06A8  1332 ; THE DIAGNOSTIC PROGRAM.
                                       06A8  1333 ;-
                                       06A8  1334
                                       06A8  1335 200$:
                             03    BB   06A8  1336          PUSHR   #^M<R0,R1>                  ; SAVE THESE
                          009D     30   06AA  1337          BSBW    RBREAK_OUT                  ; REMOVE BREAKPOINTS/PUT OPCODES BACK
                             03    BA   06AD  1338          POPR    #^M<R0,R1>                  ; RESTORE THEM
                00000014'EF        D4   06AF  1339          CLRL    DS$AA_BPTADDR               ; DON'T RESTORE NULL BREAKPOINT
             5C 0000005C'EF        7D   06B5  1340          MOVQ    L^NEW_AP_FP,AP              ; SET NEW AP/FP                      [12]
             5E 00000064'EF        D0   06BC  1341          MOVL    L^NEW_SP,SP                 ; JAM A NEW POINTER INTO 'SP'        [12]
             7E 00000068'EF        7D   06C3  1342          MOVQ    L^NEW_PC_PSL,-(SP)          ; SETUP FOR REI                      [12]
                   04 AE    10     C8   06CA  1343          BISL    #PSL$M_TBIT,4(SP)           ; SET THE T-BIT                      [12]
                00000000'EF   01    88   06CE  1344          BISB2   #1@V_TBIT,L^DEBUG$GB_FLAGS  ; SET T-BIT EXPECTED FLAG         [12]
                                      02   06D5  1345          REI                                 ; RETURN TO PROGRAM
```

```
                        06D6  1347          .SBTTL  TBIT CONDITION SUBROUTINE
                        06D6  1348 ;++
                        06D6  1349 ; FUNCTIONAL DESCRIPTION:
                        06D6  1350 ;
                        06D6  1351 ;
                        06D6  1352 ; CALLING SEQUENCE:
                        06D6  1353 ;
                        06D6  1354 ;         BSBW    DBG_TBIT
                        06D6  1355 ;
                        06D6  1356 ; INPUT PARAMETERS:      NONE
                        06D6  1357 ;
                        06D6  1358 ; IMPLICIT INPUTS:       NONE
                        06D6  1359 ;
                        06D6  1360 ; OUTPUT PARAMETERS:     NONE
                        C6D6  1361 ;
                        06D6  1362 ; IMPLICIT OUTPUTS:      NONE
                        06D6  1363 ;
                        06D6  1364 ; COMPLETION CODES:      NONE
                        06D6  1365 ;
                        06D6  1366 ; SIDE EFFECTS:          NONE
                        06D6  1367 ;--
```

```
                                06D6  1369 DBG_TBIT:
          50   0000'8F    3C    06D6  1370          MOVZWL  #SS$_RESIGNAL,R0        ; PREPARE FOR NOT WANTING THIS
    66 00000000'EF   00    E5    06DB  1371          BBCC    #V_TBIT,L^DEBUG$GB_FLAGS,40$  ; EXIT IF NOT OUR T-BIT  [12]
                                06E3  1372
                   008F    30    06E3  1373          BSBW    RBREAK_IN               ; RE-INSERT ALL BREAKPOINTS
                                06E6  1374
       00000000'EF   01    E5    06E6  1375          BBCC    #V_STEP,L^DEBUG$GB_FLAGS -    ;  [12]
                           48    06ED  1376            , 20$                         ; FINISH UP IF NOT STEPPING
             50   04 AC    D0    06EE  1377          MOVL    4(AP),R0                ; GET ADDRESS OF SIGNAL ARGS
                     7E    DF    06F2  1378          PUSHAL  -(SP)                   ; STORE CONTENTS HERE
                  08 A0    DD    06F4  1379          PUSHL   8(R0)                   ; CURRENT PC
                     04    DD    06F7  1380          PUSHL   #4                      ; FETCH 4 BYTES
       000007C3'EF   03    FB    06F9  1381          CALLS   #3,FETCH_STORE          ; GET CONTENTS
                                0700  1382
             50   04 AC    D0    0700  1383          MOVL    4(AP),R0                ; GET ADDRESS OF SIGNAL ARGS AGAIN
                  08 A0    DD    0704  1384          PUSHL   8(R0)                   ; PUT PC ON STACK
          0000009C'EF    9F    0707  1385          PUSHAB  L^FMTSTEP               ; ADDRESS OF TEXT  [12]
       00000000'9F    03    FB    070D  1386          CALLS   #3,@#DS$PRINTF          ; PRINT 'PC: CONTENTS'
                                0714  1387
       50   00000000'EF    DE    0714  1388          MOVAL   L^DS$GL_CLIBASE,R0      ; BASE CLI DATA  [12]
               03 1C A0    F5    0718  1389          SOBGTR  CLI$L_DATA(R0),10$      ; LAST PASS?
                     FF2D  31    071F  1390          BRW     CALL_CLI                ; ENTER COMMAND INTERPRETER
                                0722  1391 10$:
                   0025    30    0722  1392          BSBW    RBREAK_OUT              ; RESTORE OPCODES
       00000000'EF   03    88    0725  1393          BISB    #M_TBIT!M_STEP,L^DEBUG$GB_FLAGS ; SET STEPPING AND T-BIT EXPECT [12]
             50   04 AC    D0    072C  1394          MOVL    4(AP),R0                ; GET ADDRESS OF SIGNAL ARRAY
             0C A0    10    C8    0730  1395          BISL    #PSL$M_TBIT, 12(R0)     ; SET THE T-BIT
                     0C    11    0734  1396          BRB     30$                     ; EXIT HANDLED
                                0736  1397 20$:
             50   04 BC    DE    0736  1398          MOVAL   @4(AP),R0               ; GET ADDRESS OF SIGNAL ARRAY
    0C A0    40000010 8F    CA    073A  1399          BICL    #PSL$M_TBIT!PSL$M_TP,12(R0) ; CLEAR T-BIT(S)
                                0742  1400 30$:
       50   00000000'8F    D0    0742  1401          MOVL    #SS$_CONTINUE,R0        ; INDICATE 'HANDLED''
                                0749  1402 40$:
                           05    0749  1403          RSB                             ; RETURN
```

```
                     074A  1405          .SBTTL  TEMPORARY BREAKPOINT REMOVAL SUBROUTINE
                     074A  1406 ;++
                     074A  1407 ; FUNCTIONAL DESCRIPTION:
                     074A  1408 ;
                     074A  1409 ;       REMOVES KNOWN BREAKPOINTS
                     074A  1410 ;
                     074A  1411 ; CALLING SEQUENCE:
                     074A  1412 ;
                     074A  1413 ;       BSBW    RBREAK_OUT              NO PARAMETERS
                     074A  1414 ;
                     074A  1415 ; INPUT PARAMETERS:     NONE
                     074A  1416 ;
                     074A  1417 ; IMPLICIT INPUTS:
                     074A  1418 ;
                     074A  1419 ;       DS$AA_BPTADDR                   ; ARRAY OF KNOWN BREAKPOINTS
                     074A  1420 ;       DS$AB_BPTINST                   ; ARRAY OF SAVED 'UPCODES'
                     074A  1421 ;
                     074A  1422 ; OUTPUT PARAMETERS:    NONE
                     074A  1423 ;
                     074A  1424 ; IMPLICIT OUTPUTS:     NONE
                     074A  1425 ;
                     074A  1426 ; COMPLETION CODES:     NONE
                     074A  1427 ;
                     074A  1428 ; SIDE EFFECTS:         NONE
                     074A  1429 ;--
                     074A  1430
                     074A  1431 RBREAK_OUT:
          04   BB    074A  1432          PUSHR   #^M<R2>                 ; PRESERVE A TEMPORARY
        52   0F   D0  074C  1433          MOVL    #BPTMAX,R2              ; SET UP TABLE LENGTH
00000014'EF42   D5    074F  1434 10$:     TSTL    L^DS$AA_BPTADDR[R2]     ; SCAN FOR KNOWN BREAKPOINTS    [12]
          17   13    0756  1435          BEQL    20$                     ; SKIP IF NOT SET
                     0758  1436                                          ;
00000014'EF42   DD    0758  1437          PUSHL   L^DS$AA_BPTADDR[R2]     ; ADDRESS TO STORE ORIGINAL OPCODE  [12]
00000001'EF42   9F    075F  1438          PUSHAB  L^DS$AB_BPTINST[R2]     ; ADDRESS OF DATA TO STORE      [12]
          01   DD    0766  1439          PUSHL   #1                      ; ONLY 1 BYTE
000007C3'EF   03   FB    0768  1440          CALLS   #3,FETCH_STORE          ; STORE ORIGINAL OPCODE BACK
                     076F  1441 20$:
        DD 52   F4    076F  1442          SOBGEQ  R2,10$                  ; LOOP UNTIL DONE (INCLUDING [0])
          04   BA    0772  1443          POPR    #^M<R2>                 ; RESTORE TEMPORARY
               05    0774  1444          RSB                             ; RETURN
```

```
                     0775  1446          .SBTTL  BREAKPOINT REPLACEMEMT SUBROUTINE
                     0775  1447 ;++
                     0775  1448 ; FUNCTIONAL DESCRIPTION:
                     0775  1449 ;
                     0775  1450 ;       RESTORES ALL KNOWN BREAKPOINTS
                     0775  1451 ;
                     0775  1452 ; CALLING SEQUENCE:
                     0775  1453 ;
                     0775  1454 ;       BSBW    RBREAK_IN
                     0775  1455 ;
                     0775  1456 ; INPUT PARAMETERS:
                     0775  1457 ;
                     0775  1458 ;       NONE
                     0775  1459 ;
                     C775  1460 ; IMPLICIT INPUTS:
                     0775  1461 ;
                     0775  1462 ;       DS$AA_BPTADDR                   ; ARRAY OF KNOWN BREAKPOINTS
                     0775  1463 ;       DS$AB_BPTINST                   ; ARRAY OF SAVED 'OPCODES'
                     0775  1464 ;
                     0775  1465 ; OUTPUT PARAMETERS:    NONE
                     0775  1466 ;
                     0775  1467 ; IMPLICIT OUTPUTS:     NONE
                     0775  1468 ;
                     0775  1469 ; COMPLETION CODES:     NONE
                     0775  1470 ;
                     0775  1471 ; SIDE EFFECTS:         NONE
                     0775  1472 ;--
                     0775  1473
                     0775  1474 RBREAK_IN:                              ;
                     0775  1475          PUSHR   #^M<R2>                ; PRESERVE A TEMPORARY
                     0777  1476                                         ;
                     0777  1477          MOVL    #BPTMAX,R2             ; SET UP TABLE LENGTH
                     077A  1478 10$:      TSTL    L^DS$AA_BPTADDR[R2]    ; SCAN FOR KNOWN BREAKPOINTS            [12]
                     0781  1479          BEQL    30$                    ; SKIP IF NOT SET                       [12]
                     0783  1480                                         ;                                       [12]
                     0783  1481          PUSHAB  L^DS$AB_BPTINST[R2]    ; ADDRESS TO STORE ORIGINAL OPCODE      [12]
                     078A  1482          PUSHL   L^DS$AA_BPTADDR[R2]    ; ADDRESS TO INSERT BREAKPOINT          [12]
                     0791  1483          PUSHL   #1                     ; INSERT ONLY ONE BYTE
                     0793  1484          CALLS   #3,FETCH_STORE         ; SAVE OPCODE
                     079A  1485          BLBC    R0,20$                 ; CAN'T READ, REMOVE BREAKPOINT
                     079D  1486                                         ;
                     079D  1487          PUSHL   L^DS$AA_BPTADDR[R2]    ; ADDRESS TO INSERT BREAKPOINT          [12]
                     07A4  1488          PUSHAB  L^BPT_LITERAL          ; BPT INSTRUCTION                       [12]
                     07AA  1489          PUSHL   #1                     ; INSERT 1 BYTE
                     07AC  1490          CALLS   #3,FETCH_STORE         ; STORE IT
                     07B3  1491          BLBS    R0,30$                 ; OK, NEXT
                     07B6  1492 20$:                                    ;
                     07B6  1493          CLRL    L^DS$AA_BPTADDR[R2]    ; REMOVE BREAKPOINT                     [12]
                     07BD  1494 30$:      SOBGEQ  R2,10$                 ; LOOP UNTIL DONE (INCLUDING [0])
                     07C0  1495                                         ;
                     07C0  1496          POPR    #^M<R2>                ; RESTORE TEMPORARY
                     07C2  1497          RSB                            ; RETURN
```

```
                        07C3  1499          .SBTTL  FETCH_STORE     COPY BYTE ROUTINE
                        07C3  1500 ;++
                        07C3  1501 ; FUNCTIONAL DESCRIPTION:
                        07C3  1502 ;
                        07C3  1503 ;       This routine will copy bytes of memory from one place to another.
                        07C3  1504 ;
                        07C3  1505 ; CALLING SEQUENCE:
                        07C3  1506 ;
                        07C3  1507 ;       CALLG   (AP),FETCH_STORE
                        07C3  1508 ;
                        07C3  1509 ; INPUT PARAMETERS:
                        07C3  1510 ;
                        07C3  1511 ;        4(AP)  Count of bytes to store
                        07C3  1512 ;        8(AP)  Address to fetch data from
                        07C3  1513 ;       12(AP)  Address to store data
                        07C3  1514 ;
                        07C3  1515 ; IMPLICIT INPUTS:      NONE
                        07C3  1516 ;
                        07C3  1517 ; OUTPUT PARAMETERS:    NONE
                        07C3  1518 ;
                        07C3  1519 ; IMPLICIT OUTPUTS:     NONE
                        07C3  1520 ;
                        07C3  1521 ; COMPLETION CODES:
                        07C3  1522 ;
                        07C3  1523 ;       SS$_NORMAL       If all went well
                        07C3  1524 ;       DS$_MACHCK       If bad addresses
                        07C3  1525 ;       SS$_ACCVIO       If memory managment faults
                        07C3  1526 ;       DS$_TRANSL       If translation not valid fault
                        07C3  1527 ;
                        07C3  1528 ; SIDE EFFECTS:
                        07C3  1529 ;
                        07C3  1530 ;       Memory as specified by 12(AP) is changed
                        07C3  1531 ;--
                        07C3  1532          $OFFSET 0,NEGATIVE,<-
                        07C3  1533                  <DATA,8>, -             ; Space to save data
                        07C3  1534                  <ARG1,6*4>, -           ; Space for SETPRT arglist
                        07C3  1535                  <ADR1,8>, -             ; Descriptor of page(s)
                        07C3  1536                  <ARG2,6*4>, -           ; Space for SETPRT arglist
                        07C3  1537                  <ADR2,8>, -             ; Descriptor of page(s)
                        07C3  1538                  <LOCAL,0>>              ; Length of local storage required
                        07C3                .SAVE   LOCAL_BLOCK
                        07C3                .PSECT $ABS$ ABS
00000000                FE70                .=0
FFFFFFF8                0000                .BLKB   -8
                        FFF8        DATA:
                        FFE0        ARG1:
                        FFD8        ADR1:
                        FFC0        ARG2:
                        FFB8        ADR2:
                        FFB8        LOCAL:
```

```
                     003C  07C3  1540 .ENTRY  FETCH_STORE,^M<R2,R3,R4,R5>
                           07C5  1541
          55    5D   D0  07C5  1542          MOVL    FP,R5                       ; Base of local storage
        5E    B8 AD   9E  07C8  1543          MOVAB   LOCAL(FP),SP                ; Allocate space for local storage
             F0 AD   D4  07CC  1544          CLRL    ARG1+SETPRT$_PROT(FP)       ; Mark protection not changed
             D0 AD   D4  07CF  1545          CLRL    ARG2+SETPRT$_PROT(FP)       ; Mark protection not changed
                           07D2  1546
    6D   0000083F'EF   9E  07D2  1547          MOVAB   L^FS_HANDLR,(FP)            ; Set exception handler                 [12]
        52    04 AC   7D  07D9  1548          MOVQ    4(AP),R2                    ; Get Length and source address
           51    53   D0  07DD  1549          MOVL    R3,R1                       ; Desired address
                           07E0  1550
        63    52    00   0C  07E0  1551          PROBER  #PSL$C_KERNEL,R2,(R3)       ; Readable?
                    06   12  07E4  1552          BNEQ    10$                         ; Branch if readable
              50   00'   D0  07E6  1553          MOVL    S^#PRT$C_UR,R0              ; Desired protection
                 0CE2   30  07E9  1554          BSBW    FS_SETUP                    ; Setup and set protections
                           07EC  1555
        53    08 AC   D0  07EC  1556 10$:     MOVL    8(AP),R3                    ; Get source address
        54    F8 AD   9E  07F0  1557          MOVAB   DATA(FP),R4                 ; Intermediate
                 0023   30  07F4  1558          BSBW    FS_COPY                     ; Copy from (R3)+ to (R4)+ by R2
                 016A   30  07F7  1559          BSBW    FS_RESET                    ; Put input protections back
                           07FA  1560
        54    0C AC   D0  07FA  1561          MOVL    12(AP),R4                   ; Output address
           51    54   D0  07FE  1562          MOVL    R4,R1                       ; Desired address
        64    52    C0   0D  0801  1563          PROBEW  #PSL$C_KERNEL,R2,(R4)       ; Writeable?
                    06   12  0805  1564          BNEQ    20$                         ; YES, DO IT
              50   00'   D0  0807  1565          MOVL    S^#PRT$C_UW,R0              ; Desired protection
                 00C1   30  080A  1566          BSBW    FS_SETUP                    ; Setup and set protections
                           080D  1567
    53    F8 AD   9E  080D  1568 20$:     MOVAB   DATA(FP),R3                 ; Source for copy
                 07   10  0811  1569          BSBB    FS_COPY                     ; Copy data to target
                 014E   30  0813  1570          BSBW    FS_RESET                    ; Put output protections back
           50    01   D0  0816  1571          MOVL    S^#SS$_NORMAL,R0            ; Success!
                      04  0819  1572          RET
```

```
                                081A  1574 ;+
                                081A  1575 ; Copy data from one place to another, respect desired length Q,L,W,B
                                081A  1576 ; Read every address so we get machine checks instead of write timeouts
                                081A  1577 ;       R1 = address to report if error
                                081A  1578 ;       R2 = length 0-0F
                                081A  1579 ;       R3 = source address
                                081A  1580 ;       R4 = destination address
                                081A  1581 ;-
                                081A  1582 FS_COPY:
                06 52   03  E1  081A  1583          BBC     #3,R2,10$           ; Branch if NOT quadword
                   64   64  7D  081E  1584          MOVQ    (R4),(R4)           ; Read to check for accessability
                   84   83  7D  0821  1585          MOVQ    (R3)+,(R4)+         ; Copy quadword
                05 52   02  E1  0824  1586 10$:     BBC     #2,R2,20$           ; Branch if NOT longword
                   64       D5  0828  1587          TSTL    (R4)                ; Check accessability
                   84   83  D0  082A  1588          MOVL    (R3)+,(R4)+         ; Copy longword
                05 52   01  E1  082D  1589 20$:     BBC     #1,R2,30$           ; Branch if NOT word
                   64       B5  0831  1590          TSTW    (R4)                ; Check accessability
                   84   83  B0  0833  1591          MOVW    (R3)+,(R4)+         ; Copy word
                     05 52  E9  0836  1592 30$:     BLBC    R2,40$              ; Branch if NOT byte
                   64       95  0839  1593          TSTB    (R4)                ; Check accessability
                   84   83  90  083B  1594          MOVB    (R3)+,(R4)+         ; Copy byte
                            05  083E  1595 40$:     RSB
                                083F  1596
```

```
                                083F  1598 ;+
                                083F  1599 ; Exception handler for FETCH_STORE and FETCH_STOR_PREG routines.
                                083F  1600 ;-
                                083F  1601 FS_HANDLR:
                          0020  083F  1602          .WORD   ^MR5                    ; Need this register for old FP
          50   04 AC     7D    0841  1603          MOVQ    4(AP),R0                ; Get address of mech/signal args
          55   04 A1     D0    0845  1604          MOVL    4(R1),R5                ; Get FP at time of exception
 04 A0  00660088 8F     D1    0849  1605          CMPL    #DS$_MCHK,4(R0)         ; MACHINE CHECK?
                  0D     12    0851  1606          BNEQ    110$                    ; NO, TRY NEXT
    50  00000162'EF     9E    0853  1607          MOVAB   L^T_MACHCK,R0           ; POINT TO ERROR TEXT                   [12]
          51   10 A1     D0    085A  1608          MOVL    16(R1),R1               ; GET VIRTURAL ADDRESS
                  45     11    085E  1609          BRB     150$                    ; AND PRINT IT
 04 A0  006600A0 8F     D1    0860  1610 110$:     CMPL    #DS$_TRANSL,4(R0)       ; TRANSLATION NOT VALID?
                  0D     12    0868  1611          BNEQ    120$                    ; NO, TRY NEXT
          51   0C A0     D0    086A  1612          MOVL    8+4(R0),R1              ; Get virtural address
    50  00000197'EF     9E    086E  1613          MOVAB   L^T_TRANSL,R0           ; POINT TO ERROR TEXT                   [12]
                  2E     11    0875  1614          BRB     150$                    ; PRINT
 04 A0  00000000'8F     D1    0877  1615 120$:     CMPL    #SS$_ACCVIO,4(R0)       ; ACCESS VIOLATION?
                  0D     12    087F  1616          BNEQ    130$                    ; No, try next
          51   0C A0     D0    0881  1617          MOVL    8+4(R0),R1              ; Get virtural address
    50  00000183'EF     9E    0885  1618          MOVAB   L^T_ACCVIO,R0           ; POINT TO TEXT                         [12]
                  17     11    088C  1619          BRB     150$                    ; PRINT
                                088E  1620 130$:
 04 A0  00000000'8F     D1    088E  1621          CMPL    #SS$_ROPRAND,4(R0)      ; Invalid operand? (must be IPR if so)
                  07     12    0896  1622          BNEQ    145$                    ; Nope.
       25 0C A1    00     E4    0898  1623          BBSC    #0,12(R1),155$          ; Clear success bit in return code
                  23     11    089D  1624          BRB     155$                    ; Exit
                                089F  1625 145$:
  50    0000'8F     3C    089F  1626          MOVZWL  #SS$_RESIGNAL,R0        ; PAWN IT OFF ON SOMEONE ELSE
                  04     08A4  1627          RET                             ; RETURN FROM CONDITION HANDLER
                                08A5  1628
                                08A5  1629 150$:     $PRINTI_S L^FMT_EXCEPTION,R0,R1 ; TYPE THE ERROR TEXT                   [12]
    50   04 AC     7D    08B6  1630          MOVQ    4(AP),R0                ; SIGNAL AND MECHANISM ADDRESSES
 0C A1   04 A0     D0    08BA  1631          MOVL    4(R0),12(R1)            ; Put condition in R0
          00A2     30    08BF  1632          BSBW    FS_RESET                ; Put protections back
                         08C2  1633 155$:     $UNWIND_S                       ; Return to caller
                  04     08CD  1634          RET                             ; RETURN
```

```
                                08CE  1636 ;+
                                08CE  1637 ; Set protections for page(s)
                                08CE  1638 ;       R0 = desired protection
                                08CE  1639 ;       R1 = desired address, preserved
                                08CE  1640 ;       R2 = length of area, 0-F
                                08CE  1641 ;-
                                08CE  1642 FS_SETUP:
                          02    BB  08CE  1643          PUSHR   #^M<R1>                 ; Save address
                      F0 AD   50    9A  08D0  1644          MOVZBL  R0,ARG1+SETPRT$_PROT(FP); Set desired protection
                      D0 AD   50    9A  08D4  1645          MOVZBL  R0,ARG2+SETPRT$_PROT(FP); Set desired protection
                   50   01FF 8F    3C  08D8  1646          MOVZWL  #^X1FF,R0               ; Byte within page bits
                D8 AD   51   50    CB  08DD  1647          BICL3   R0,R1,ADR1(FP)          ; Starting address for this page
                   51   FF A142   9E  08E2  1648          MOVAB   -1(R1)[R2],R1           ; Address of last byte to be affected
                DC AD   51   50    CB  08E7  1649          BICL3   R0,R1,ADR1+4(FP)        ; Ending address
                  DC AD    D8 AD   D1  08EC  1650          CMPL    ADR1(FP),ADR1+4(FP)     ; Start same as end?
                           28      13  08F1  1651          BEQL    20$                     ; Branch if only one page need modifications
                                      08F3  1652
                  B8 AD    DC AD   D0  08F3  1653          MOVL    ADR1+4(FP),ADR2(FP)     ; Set start of page
                  BC AD    B8 AD   D0  08F8  1654          MOVL    ADR2(FP),ADR2+4(FP)     ; Set end of page
                  D4 AD    D2 AD   DE  08FD  1655          MOVAL   ARG2+SETPRT$_PROT+2(FP),-
                                      0902  1656                  ARG2+SETPRT$_PRVPRT(FP) ; Address to return protection
                           C8 AD   7C  0902  1657          CLRQ    ARG2+SETPRT$_RETADR(FP) ; No return address and kernel mode
                  C4 AD    B8 AD   7E  0905  1658          MOVAQ   ADR2(FP),-
                                      090A  1659                  ARG2+SETPRT$_INADR(FP)  ; Address of start and end pages
                     C0 AD    05    D0  090A  1660          MOVL    #SETPRT$_NARGS,ARG2(FP) ; Set count of args
                                      090E  1661
           00000000'EF    C0 AD   FA  090E  1662          CALLG   ARG2(FP),SYS$SETPRT     ; Set the protection
                          2E 50   E9  0916  1663          BLBC    R0,30$                  ; Branch if it failed
                  D0 AD   40 8F   88  0919  1664          BISB    #64,ARG2+SETPRT$_PROT(FP); Flag protection changed
                                      091E  1665
                  DC AD    D8 AD   D0  091E  1666 20$:     MOVL    ADR1(FP),ADR1+4(FP)     ; Set end address of range
                  F4 AD    F2 AD   DE  0923  1667          MOVAL   ARG1+SETPRT$_PROT+2(FP),-
                                      0928  1668                  ARG1+SETPRT$_PRVPRT(FP) ; Address to return protection
                           E8 AD   7C  0928  1669          CLRQ    ARG1+SETPRT$_RETADR(FP) ; No return address and kernel mode
                  E4 AD    D8 AD   7E  092B  1670          MOVAQ   ADR1(FP),-
                                      0930  1671                  ARG1+SETPRT$_INADR(FP)  ; Address of start and end pages
                     E0 AD    05    D0  0930  1672          MOVL    #SETPRT$_NARGS,ARG1(FP) ; Set count of args
                                      0934  1673
           00000000'EF    E0 AD   FA  0934  1674          CALLG   ARG1(FP),SYS$SETPRT     ; Set the protection
                          08 50   E9  093C  1675          BLBC    R0,30$                  ; Branch if it failed
                  F0 AD   40 8F   88  093F  1676          BISB    #64,ARG1+SETPRT$_PROT(FP); Flag protection changed
                             02   BA  0944  1677          POPR    #^M<R1>                 ; Restore R1
                                  05  0946  1678          RSB
                                      0947  1679
                       52    50   D0  0947  1680 30$:     MOVL    R0,R2                   ; Save SETPRT return code
                                      094A  1681                                          ; Address already on stack
                00000183'EF    9F  094A  1682          PUSHAB  L^T_ACCVIO              ; ADDRESS OF ACCESS VIOLATION INSERT    [12]
                00000180'EF    9F  0950  1683          PUSHAB  L^FMT_EXCEPTION         ; FAO STRING                            [12]
           00000000'EF    03    FB  0956  1684          CALLS   #3, DSX$PRINTI          ; TYPE ACCESS VIOLATION TEXT
                                      095D  1685
                     0004   30  095D  1686          BSBW    FS_RESET                ; Reset the protections
                  50   52    D0  0960  1687          MOVL    R2,R0                   ; Restore return code
                             04  0963  1688          RET                             ; RETURN TO CALLER OF FETCH_STORE
                                 0964  1689 ;+
                                 0964  1690 ; Routine to restore protections on pages changed
                                 0964  1691 ;       R5 = FP of local data structure
                                 0964  1692 ;-
```

```
                                0964  1693 FS_RESET:
            50    F0 A5   9E    0964  1694          MOVAB   ARG1+SETPRT$_PROT(R5),R0; Address of protections
            0C 60   06    E5    0968  1695          BBCC    #6,(R0),10$             ; Branch if protection not changed
            60    02 A0   90    096C  1696          MOVB    2(R0),(R0)              ; Put old protection in arglist
00000000'EF     E0 A5   FA    0970  1697          CALLG   ARG1(R5),SYS$SETPRT     ; Set protection back
                              0978  1698
            50    D0 A5   9E    0978  1699 10$:     MOVAB   ARG2+SETPRT$_PROT(R5),R0; Address of protections
            0C 60   06    E5    097C  1700          BBCC    #6,(R0),20$             ; Branch if protection not changed
            60    02 A0   90    0980  1701          MOVB    2(R0),(R0)              ; Put old protection in arglist
00000000'EF     C0 A5   FA    0984  1702          CALLG   ARG2(R5),SYS$SETPRT     ; Set protection back
                              098C  1703
                        05    098C  1704 20$:     RSB                             ; Return
                              098D  1705
```

```
                                098D  1707          .SBTTL  Handle EXAMINE/DEPOSIT of IPRs in KERNEL mode
                                098D  1708 ;+
                                098D  1709 ; CHMK handler: gets control via CHMK dispatch routine.  It calls routine
                                098D  1710 ; FETCH_STOR_PREG, in KERNEL mode, to allow use of MFPR/MTPR instructions
                                098D  1711 ; and handling of possible exception due to access
                                098D  1712 ;       PARAMETERS:
                                098D  1713 ;               R1      Value to deposit (if function is DEPOSIT)
                                098D  1714 ;               R3      IPR code
                                098D  1715 ;               R5      function code: 1 if EXAMINE, 0 if DEPOSIT
                                098D  1716 ;       RETURN:
                                098D  1717 ;               R1      Value examined--is "read-back" of deposited
                                098D  1718 ;                       value if function is DEPOSIT
                                098D  1719 ;-
                                098D  1720 CMK$SGIPR::
                     51   DD    098D  1721          PUSHL   R1                      ; Data, if deposit (else save space)
                     6E   DF    098F  1722          PUSHAL  (SP)                    ; Address of where to put/get data
                     55   DD    0991  1723          PUSHL   R5                      ; Flags word to determine if exam/dep
                     53   DD    0993  1724          PUSHL   R3                      ; Register Code
                9C'AF     03   FB  0995  1725          CALLS   #3,B^FETCH_STOR_PREG    ; Do it
                     02   BA    0999  1726          POPR    #^M1                    ; Get data back if examine (else clear stack
                          02    099B  1727          REI                             ; Return from the interrupt
                                099C  1728
```

```
                                099C  1730 .SBTTL  FETCH_STOR PREG Move data to/from IPR
                                099C  1731 ;++
                                099C  1732 ; FUNCTIONAL DESCRIPTION:
                                099C  1733 ;
                                099C  1734 ;       This routine will either copy data from memory to an internal CPU
                                099C  1735 ;       register (DEPOSIT) or move data from an internal CPU register into
                                099C  1736 ;       memory (EXAMINE).  If function is DEPOSIT, it will also preform an
                                099C  1737 ;       EXAMINE to the same IPR, providing a verify read-back.
                                099C  1738 ;
                                099C  1739 ; CALLING SEQUENCE:
                                099C  1740 ;
                                099C  1741 ;       CALLS   #3,FETCH_STOR_PREG
                                099C  1742 ;
                                099C  1743 ; INPUT PARAMETERS:
                                C99C  1744 ;
                                099C  1745 ;       4(AP)   The code for the desired IPR (range 0 to FF(X))
                                099C  1746 ;       8(AP)   A flag (=1 for EXAMINE, =0 for DEPOSIT)
                                099C  1747 ;       12(AP)  Address of location to get/store data
                                099C  1748 ;
                                099C  1749 ; IMPLICIT INPUTS:      None
                                099C  1750 ;
                                099C  1751 ; OUTPUT PARAMETERS:    None
                                099C  1752 ;
                                099C  1753 ; IMPLICIT OUTPUTS:     None
                                099C  1754 ;
                                099C  1755 ; COMPLETION CODES:
                                099C  1756 ;
                                099C  1757 ;       R0 bits define operations completed:
                                099C  1758 ;               bit0 set = no error
                                099C  1759 ;               bit1 set = DEPOSIT completed with no error
                                099C  1760 ;
                                099C  1761 ; SIDE EFFECTS:         None
                                099C  1762 ;--
                                099C  1763
                          000C  099C  1764 .ENTRY  FETCH_STOR_PREG,^M<R2,R3>
6D  FFFFFE9B EF     9E        099E  1765          MOVAB   L^FS_HANDLR,(FP)        ; Set up exception handler              [12]
    52    04 AC     D0        09A5  1766          MOVL    4(AP),R2                ; Get IPR code
    53    0C AC     D0        09A9  1767          MOVL    12(AP),R3               ; Get memory address
       50    01     D0        09AD  1768          MOVL    #1,R0                   ; Presume success
     07 08 AC     E8        09B0  1769          BLBS    8(AP),10$               ; EXAMINE only
       52    63     DA        09B4  1770          MTPR    (R3),R2                 ; Attempt to move data to register
    00 50    01     E2        09B7  1771          BBSS    #1,R0,10$               ; DEPOSIT worked--now do read-back
       63    52     DB        09BB  1772 10$:     MFPR    R2,(R3)                 ; Try to move data from register
                     04        09BE  1773          RET
```

```
                                09BF  1775 .SBTTL  DSV$DEBUG - Process DEBUG Command                                [14]
                                09BF  1776 ;++                                                                        [14]
                                09BF  1777 ; FUNCTIONAL DESCRIPTION                                                   [14]
                                09BF  1778 ;       This routine will process the DEBUG command by allocating          [14]
                                09BF  1779 ;       memory space for the debugger, loading the debugger, and           [14]
                                09BF  1780 ;       then calling it.                                                   [14]
                                09BF  1781 ;                                                                          [14]
                                09BF  1782 ; CALLING SEQUENCE                                                         [14]
                                09BF  1783 ;       BSBW    DSV$DEBUG                                                  [14]
                                09BF  1784 ;                                                                          [14]
                                09BF  1785 ; INPUT PARAMETERS                                                         [14]
                                09BF  1786 ;       NONE                                                               [14]
                                09BF  1787 ;                                                                          [14]
                                09BF  1788 ; OUTPUT PARAMETERS                                                        [14]
                                09BF  1789 ;       NONE                                                               [14]
                                09BF  1790 ;                                                                          [14]
                                09BF  1791 ;--                                                                        [14]
                                09BF  1792
                    00000002    09BF  1793 TRANS_VECTOR_OFFSET = 2                ;BYTE OFFSET FROM TOP OF IMAGE   [14]
                                09BF  1794                                        ;   HEADER, CONTAINING OFFSET    [14]
                                09BF  1795                                        ;   TO TRANSFER VECTOR ARRAY      [14]
                                09BF  1796
 45 58 45 2E 000009C7'010E0000' 09BF  1797 DBG_DEF:        .ASCID /.EXE/          ;                                [14]
                                09CB  1798 DSV$DEBUG::                            ;                                [14]
                         1A  E1 09CB  1799         BBC     #DSA$V_DEBUG,-         ;IF DEBUGGER ALREADY LOADED      [14]
              03 0000FE00'EF    09CD  1800                 DSA$GL_FLAGS,5$        ;THEN                            [14]
                       0103  31 09D3  1801         BRW     27$                    ;   DON'T RELOAD                 [14]
                             09D6  1802 5$:       $DS_PRINTF_S    FMT_DBG_MSG    ;PRINT MESSAGE                   [14]
                         1C  E1 09E3  1803         BBC     #DSA$V_USER,-          ;IF USER MODE                    [14]
            48 0000FE00'EF    09E5  1804                 DSA$GL_FLAGS,10$       ;THEN                            [14]
00000171'EF 000FFFFF 8F   D0 09EB  1805         MOVL    #^X000FFFFF,-          ;   SET VA FOR TOP OF DEBUGGER   [14]
                             09F6  1806                 DBG_ADDR_REQ+4         ;    (1000K)                     [14]
            00010E00 8F   C3 09F6  1807         SUBL3   #DEBUGGER_SIZE,-       ;   SUBTRACT DEBUGGER SIZE       [14]
            00000171'EF      09FC  1808                 DBG_ADDR_REQ+4,-       ;    FROM TOP ADDRESS            [14]
            0000016D'EF      0A01  1809                 DBG_ADDR_REQ           ;    AND STORE AS START VA       [14]
                             0A06  1810         $CRETVA_S       DBG_ADDR_REQ, -;   CALL SYSTEM SERVICE TO ALLO-[14]
                             0A06  1811                 DBG_ADDR_RTN           ;    CATE MEMORY FOR DEBUGGER    [14]
00000171'EF 00000175'EF   D0 0A1B  1812         MOVL    DBG_ADDR_RTN,DEBUG_LO  ;   STORE DEBUGGER LOW ADDRESS  [14]
00000181'EF 00000179'EF   D0 0A26  1813         MOVL    DBG_ADDR_RTN+4,DEBUG_HI;   STORE DEBUGGER HIGH ADDRESS [14]
                       5A 11 0A31  1814         BRB     20$                    ;ELSE                            [14]
            00010E00 8F   C3 0A33  1815 10$:     SUBL3   #DEBUGGER_SIZE,-       ;   SUBTRACT DEBUGGER SIZE       [14]
            00000000'EF      0A39  1816                 DS$GL_MEMSIZE,-        ;    FROM TOP ADDRESS            [14]
            00000175'EF      0A3E  1817                 DBG_ADDR_RTN           ;    AND STORE AS START VA       [14]
00000175'EF 00000200 8F   C2 0A43  1818         SUBL2   #512,DBG_ADDR_RTN      ;    DON'T USE TOP PAGE OF MEM.  [14]
                             0A4E  1819                                        ;    (IT'S USED BY A DIAGNOSTIC!)[14]
00000000'EF 00000200 8F   C3 0A4E  1820         SUBL3   #512,DS$GL_MEMSIZE,-   ;   TOP OF DEBUGGER AT TOP OF MEM[14]
            00000181'EF      0A59  1821                 DEBUG_HI               ;    MINUS LAST PAGE             [14]
0000017D'EF 00000175'EF   D0 0A5E  1822         MOVL    DBG_ADDR_RTN,DEBUG_LO  ;   BOTTOM OF DEBUGGER          [14]
00000185'EF 0000017D'EF   D0 0A69  1823         MOVL    DEBUG_LO,SYMTAB_HI     ;   ALSO WHERE SYM. TABLES START[14]
            00000000'EF   16 0A74  1824         JSB     MAP_DEBUGGER           ;   MAP MEMORY SPACE FOR DEBUGGER[14]
                    10 50 E8 0A7A  1825         BLBS    R0,20$                 ;   IF MAPPING ERROR THEN       [14]
                             0A7D  1826         $DS_PRINTF_S    FMT_NO_ROOM    ;       PRINT MSG               [14]
                     009D 31 0A8A  1827         BRW     DSV$DEBUG_EXIT         ;       LEAVE                   [14]
            00000175'EF   DD 0A8D  1828 20$:     PUSHL   DBG_ADDR_RTN           ;ADDRESS TO LOAD DEBUGGER        [14]
            00010E00 8F   DD 0A93  1829         PUSHL   #DEBUGGER_SIZE         ;LENGTH OF LOAD AREA             [14]
            FFFFFF20 EF   7F 0A99  1830         PUSHAQ  L^DBG_DEF              ;ADDRESS OF FILE DEFAULTS        [14]
                  08 A2   7F 0A9F  1831         PUSHAQ  CLI$Q_FILE(R2)         ;ADDRESS OF FILE DESCRIPTOR      [14]
```

```
                00000000'EF   04   FB  0AA2  1832          CALLS   #4,L^DSX$LOAD                   ;LOAD THE DEBUGGER                      [14]
                00000000'EF   16   0AA9  1833          JSB     DSR$COMPLETION                  ;TYPE ERROR TEXT IF NECESSARY           [14]
                      09 50   E8   0AAF  1834          BLBS    R0,25$                          ;IF ERROR THEN                          [14]
                00000000'EF   16   0AB2  1835          JSB     UNMAP_DEBUGGER                  ;   UNMAP MEMORY WE JUST ALLOC'D        [14]
                      006F    31   0AB8  1836          BRW     DSV$DEBUG_EXIT                  ;   LEAVE                                 [14]
                              0ABB  1837 25$:     $DS_PRINTF_S    FMT_DBG_LOAD,-              ;PRINT MESSAGE                          [14]
                              0ABB  1838                      DBG_ADDR_RTN                    ;                                       [14]
                04000000 8F   C8   0ACE  1839          BISL    #1@DSA$V_DEBUG,-                ;SET DEBUGGER RESIDENT FLAG             [14]
                0000FE00'EF         0AD4  1840                  DSA$GL_FLAGS                    ;                                       [14]
                         0E   BB   0AD9  1841 27$:     PUSHR   #^M<R1,R2,R3>                   ;SAVE REGISTERS                         [14]
          51    80000000 8F   D0   0ADB  1842          MOVL    #^X80000000,R1                  ;SET DS_ENVIRON BIT                     [14]
                         1C   E1   0AE2  1843          BBC     #DSA$V_USER,-                   ;IF USER MODE                           [14]
            07 0000FE00'EF         0AE4  1844                  DSA$GL_FLAGS,30$                ;THEN                                   [14]
          51    40000000 8F   C8   0AEA  1845          BISL    #1@30,R1                        ;   SET USER MODE BIT                   [14]
                         51   DD   0AF1  1846 30$:     PUSHL   R1                              ;CLI STATUS BITS                        [14]
                         00   DD   0AF3  1847          PUSHL   #0                              ;NULL ARGUMENT                          [14]
                00000000'EF   DF   0AF5  1848          PUSHAL  DIAG_FILE_NAME                  ;PASS ADDR. OF COUNTED STRING           [14]
                00000000'EF   DF   0AFB  1849          PUSHAL  IMAGE_HEADER                    ;PASS ADDR. OF IMAGE HEADER             [14]
                         00   DD   0B01  1850 50$:     PUSHL   #0                              ;NULL ARGUMENT                          [14]
                00000B2B'EF   DF   0B03  1851          PUSHAL  TRANS_VECTOR                    ;TRANSFER ADDRESS VECTOR                [14]
          51    00000175'EF   D0   0B09  1852          MOVL    DBG_ADDR_RTN,R1                 ;PUT START OF DEBUGGER IMAGE            [14]
                              0B10  1853                                                      ; HEADER IN R1                          [14]
            52    02 A1   9A   0B10  1854          MOVZBL  TRANS_VECTOR_OFFSET(R1),R2;GET OFFSET TO TRANSFER VECTOR[14]
          53    52   51   C1   0B14  1855          ADDL3   R1,R2,R3                        ;GET ADDRESS OF VECTOR                  [14]
                53   04   C0   0B18  1856          ADDL2   #4,R3                           ;GET SECOND ENTRY                       [14]
                51   63   C0   0B1B  1857          ADDL2   (R3),R1                         ;MAKE TRANSFER ADDRESS ABSOLUTE         [14]
                61   06   FB   0B1E  1858          CALLS   #6,(R1)                         ;CALL THE DEBUGGER                      [14]
                              0B21  1859 CONT_FROM_DEBUG:                                   ;                                       [14]
                         0E   BA   0B21  1860          POPR    #^M<R1,R2,R3>                   ;RESTORE REGISTERS                      [14]
          5E    00000058 8F   C0   0B23  1861          ADDL2   #<22*4>,SP                      ;RESTORE STACK (DEBUGGER MESSED         [14]
                              0B2A  1862                                                      ; IT UP).                               [14]
                              0B2A  1863 DSV$DEBUG_EXIT:                                    ;                                       [14]
                         05   0B2A  1864          RSB                                     ;RETURN                                 [14]
                              0B2B  1865 TRANS_VECTOR:                                      ;USED BY DEBUGGER                       [14]
                    00000000  0B2B  1866                  .LONG   0                       ;                                       [14]
                    00000B1F' 0B2F  1867                  .LONG   CONT_FROM_DEBUG-2;                                            [14]
                              0B33  1868
                              0B33  1869          .END
```

```
$$ARGS                          =  00000004     D
$$N                             =  00000002     D
$$T1                            =  00000014     D
$$T2                            =  00000007     D
$ER                             =  00000001     D
$MODULE                            00000000 R   D  03
AAP                                000002D8 R   D  03
ADR1                               FFFFFFD8     D
ADR2                               FFFFFFB8     D
AEM                                00000254 R   D  03
AFP                                000002DB R   D  03
AKM                                0000024D R   D  03
APC                                000002E1 R   D  03
APSL                               000002E4 R   D  03
APSLMNE                            000001F8 RG  D  03
AR0                                000002B2 R   D  03
AR1                                000002B5 R   D  03
AR10                               000002D0 R   D  03
AR11                               000002D4 R   D  03
AR2                                000002B8 R   D  03
AR3                                000002BB R   D  03
AR4                                000002BE R   D  03
AR5                                000002C1 R   D  03
AR6                                000002C4 R   D  03
AR7                                000002C7 R   D  03
AR8                                000002CA R   D  03
AR9                                000002CD R   D  03
ARG1                               FFFFFFE0     D
ARG2                               FFFFFFC0     D
ASM                                0000025E R   D  03
ASP                                000002DE R   D  03
AUM                                00000269 R   D  03
BASEADDRESS                        00000058 R   D  02
BIT...                          =  0000000F     D
BPTCODE                         =  00000003   G D
BPTMAX                          =  0000000F   G D
BPT_LITERAL                        00000006 R   D  03
CALL_CLI                           0000064F R   D  04
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_SIZE                         00000444     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_LAST                         00000024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEST                         00000020     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       00C0003C     D
CLI$V_ADAPTER                   =  000u0018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BREAK                     =  0000000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
CMK$SGIPR                          0000098D RG  D  04
CMK$_                           =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
COND_DEBUG                         000005DE RG  D  04
CONT_FROM_DEBUG                    00000B21 R   D  04
DATA                               FFFFFFF8     D
DBG_ADDR_REQ                       0000016D R   D  02
DBG_ADDR_RTN                       00000175 R   D  02
DBG_BREAK                          00000605 R   D  04
DBG_DEF                            000009BF R   D  04
DBG_TBIT                           000006D6 R   D  04
DEBUG$GB_FLAGS                     00000000 RG  D  02
DEBUGGER_SIZE                   =  00010E00     D
DEBUG_HI                           00000181 RG  D  02
```

```
DEBUG_LO                        0000017D RG D  02
DFLTSIZE                        00000054 R  D  02
DIAG_FILE_NAME                  ******** GX    00
DIR...                        = FFFFFFFF    D
DS$AA_BPTADDR                   00000014 RG D  02
DS$AB_BPTINST                   00000001 RG D  02
DS$CLI                          ********   X   04
DS$CVTREG                       ********   X   04
DS$GL_CLIBASE                   ********   X   04
DS$GL_MEMSIZE                   ********   X   04
DS$GL_RADIX                     ********   X   04
DS$K_ERROR                    = 00000002    D
DS$K_NORMAL                   = 00000001    D
DS$K_SEVERE                   = 00000004    D
DS$K_SUBSYS                   = 00000066    D
DS$K_WARNING                  = 00000000    D
DS$M_ABRTFLG                  = 00000040    D
DS$M_BADTIME                  = 00100000    D
DS$M_BATCH                    = 00400000    D
DS$M_BRKCLR                   = 00001000    D
DS$M_BRKPT                    = 00000800    D
DS$M_CHARFLG                  = 00000100    D
DS$M_CMDFLG                   = 00000080    D
DS$M_CTRLC                    = 00000001    D
DS$M_CTRLO                    = 00010000    D
DS$M_DEVFLG                   = 00000200    D
DS$M_DISABLCC                 = 01000000    D
DS$M_DONFLG                   = 00002000    D
DS$M_ERRFLG                   = 00000010    D
DS$M_EXCEPT                   = 00080000    D
DS$M_EXETST                   = 00040000    D
DS$M_HLTFLG                   = 00000008    D
DS$M_LODFLG                   = 00000002    D
DS$M_MEMMGT                   = 00008000    D
DS$M_OUTPUT                   = 00800000    D
DS$M_RUBFLG                   = 00000020    D
DS$M_SCRIPT                   = 00200000    D
DS$M_SETIMR                   = 02000000    D
DS$M_STRFLG                   = 00000004    D
DS$M_SUBT                     = 00004000    D
DS$M_SYSFLG                   = 00000400    D
DS$M_TIMRON                   = 00020000    D
DS$PRINTF                       ********   X   04
DS$V_ABRTFLG                  = 00000006    D
DS$V_BADTIME                  = 00000014    D
DS$V_BATCH                    = 00000016    D
DS$V_BRKCLR                   = 0000000C    D
DS$V_BRKPT                    = 0000000B    D
DS$V_CHARFLG                  = 00000008    D
DS$V_CMDFLG                   = 00000007    D
DS$V_CTRLC                    = 00000000    D
DS$V_CTRLO                    = 00000010    D
DS$V_DEVFLG                   = 00000009    D
DS$V_DISABLCC                 = 00000018    D
DS$V_DONFLG                   = 0000000D    D
DS$V_ERRFLG                   = 00000004    D
DS$V_EXCEPT                   = 00000013    D
DS$V_EXETST                   = 00000012    D
DS$V_HLTFLG                   = 00000003    D
DS$V_LODFLG                   = 00000001    D
DS$V_MEMMGT                   = 0000000F    D
DS$V_OUTPUT                   = 00000017    D
DS$V_RUBFLG                   = 00000005    D
DS$V_SCRIPT                   = 00000015    D
DS$V_SETIMR                   = 00000019    D
DS$V_STRFLG                   = 00000002    D
DS$V_SUBT                     = 0000000E    D
DS$V_SYSFLG                   = 0000000A    D
DS$V_TIMRON                   = 00000011    D
DS$_ARITH                     = 006600D0    D
DS$_ASBE                      = 00660118    D
DS$_BADLINK                   = 006600F0    D
DS$_BADTYPE                   = 006600E8    D
DS$_BIIC                      = 00660120    D
DS$_CHME                      = 006600A8    D
DS$_CHMK                      = 006600E0    D
DS$_DEVNAME                   = 00660108    D
DS$_ERROR                     = 00660002    D
DS$_FHWE                      = 00660068    D
DS$_FRAGBUF                   = 00660080    D
DS$_ICBUSY                    = 006600C8    D
DS$_ICERR                     = 006600C0    D
DS$_IHWE                      = 00660060    D
DS$_ILLCHAR                   = 00660018    D
DS$_ILLPAGCNT                 = 00660078    D
DS$_ILLUNIT                   = 00660100    D
DS$_INSFMEM                   = 00660050    D
DS$_IPL2HI                    = 006600B8    D
DS$_IVADDR                    = 00660040    D
DS$_IVVECT                    = 00660038    D
DS$_KRNLSTK                   = 00660090    D
DS$_LOGIC                     = 00660070    D
DS$_MCHK                      = 00660088    D
DS$_MMOFF                     = 00660058    D
DS$_NEEDUNIT                  = 006600F8    D
DS$_NODE                      = 00660128    D
DS$_NOPCS                     = 00660110    D
DS$_NORMAL                    = 00660001    D
DS$_NOSUPPORT                 = 006600B1    D
DS$_NOTDON                    = 00660030    D
DS$_NOTIMP                    = 006600B0    D
DS$_NULLSTR                   = 00660010    D
DS$_OVERFLOW                  = 00660008    D
DS$_POWER                     = 00660098    D
DS$_PROGERR                   = 00660020    D
DS$_SEVERE                    = 00660004    D
DS$_TRANSL                    = 006600A0    D
DS$_TRUNCATE                  = 00660028    D
DS$_UNEXPINT                  = 006600D8    D
DS$_VASFULL                   = 00660048    D
DS$_WARNING                   = 00660000    D
DSA$AL_APTMAIL                  0000FE00    D
DSA$AT_APTTXT                   0000FA00    D
DSA$GL_APTCOM                   0000FE04    D
```

```
DSA$GL_DEVLEN                    0000FE58     D
DSA$GL_ERRNO                     0000FE44     D
DSA$GL_EVENT                     0000FE48     D
DSA$GL_FLAGS                     0000FE00     D
DSA$GL_MSGTYP                    0000FE40     D
DSA$GL_PASSES                    0000FE08     D
DSA$GL_PASSNO                    0000FE54     D
DSA$GL_SECTNO                    0000FE10     D
DSA$GL_SID                       0000FE14     D
DSA$GL_SUBTNO                    0000FE4C     D
DSA$GL_TESTNO                    0000FE50     D
DSA$GL_UNITS                     0000FE0C     D
DSA$GQ_MSGPTR                    0000FE68     D
DSA$GT_DEVNAM                    0000FE5C     D
DSA$V_DEBUG                  =   0000001A     D
DSA$V_USER                   =   0000001C     D
DSP$REMOVEBREAK                  00000553 RG  D  04
DSR$COMPLETION                   ********   X    04
DSV$DEBUG                        000009CB RG  D  04
DSV$DEBUG_EXIT                   00000B2A R   D  04
DSX$LOAD                         ********   X    04
DSX$PRINTI                       ********   X    04
EXAMBUFFER                       00000083 R   D  02
EXAMQWDBUF                       0000007B R   D  02
EXAMSIZE                     =   00000020     D
FAOARG                           00000107 R   D  02
FAOL$_CTRSTR                 =   00000004     D
FAOL$_NARGS                  =   00000004     D
FAOL$_OUTBUF                 =   0000000C     D
FAOL$_OUTLEN                 =   00000008     D
FAOL$_PRMLST                 =   00000010     D
FAOLST                           0000C0F3 R   D  02
FETCH_STORE                      000007C3 RG  D  04
FETCH_STOR_PREG                  0000099C RG  D  04
FIND_BPT                         000005C7 R   D  04
FMTBPT                           00000085 R   D  03
FMTBPTADR                        000000A7 R   D  03
FMTBPTLST                        000000B2 R   D  03
FMTBPTNONE                       000000C9 R   D  03
FMTBPTNOTSET                     000000DE R   D  03
FMTDIRECTIVE                     00000070 R   D  02
FMTEXAM                          0000012B R   D  03
FMTEXAM_ASCII                    00000118 R   D  03
FMTINVLDREG                      000000F9 R   D  03
FMTNOBPTSLEFT                    00000139 R   D  03
FMTSTEP                          0000009C R   D  03
FMT_BADPREG                      000001E4 R   D  03
FMT_BASE_OUT                     00000007 R   D  03
FMT_BYTE                         00000080 R   D  03
FMT_DBG_LOAD                     00000121 R   D  02
FMT_DBG_MSG                      0000010B R   D  02
FMT_DEC                          00C00064 R   D  03
FMT_DFLT_OUT                     0000002D R   D  03
FMT_EXCEPTION                    000001B0 R   D  03
FMT_HEX                          00000058 R   D  03
FMT_LONG                         00000072 R   D  03
FMT_NO_ROOM                      0000013A R   D  02
FMT_OCT                          0000006C R   D  03
FMT_WORD                         0000007B R   D  03
FS_COPY                          0000081A R   D  04
FS_HANDLR                        0000083F R   D  04
FS_RESET                         00000964 R   D  04
FS_SETUP                         000008CE R   D  04
HTAB                         =   00000009     D
IMAGE_HEADER                     ********     X  04
INITBASE                     =   00000000     D
INITSIZE                     =   00000004     D
LOCAL                            FFFFFFB8     D
MAP_DEBUGGER                     ********     X  04
MODELST                          00000239 RG  D  03
M_STEP                       =   00000002     D
M_TBIT                       =   00000001     D
NEW_AP_FP                        0000005C R   D  02
NEW_PC_PSL                       00000068 R   D  02
NEW_SP                           00000064 R   D  02
PRT$C_UR                         ********     X  04
PRT$C_UW                         ********     X  04
PSL$C_KERNEL                 =   00000000     D
PSL$M_TBIT                   =   00000010     D
PSL$M_TP                     =   40000000     D
PSL$V_IS                     =   0000001A     D
PSLBUFFER                        000000A3 R   D  02
PSLBUFSZ                     =   00000050     D
PSLREG                       =   00000010     D
RBREAK_IN                        00000775 R   D  04
RBREAK_OUT                       0000074A R   D  04
REGNAMLST                        0000026E R   D  03
SAVABS...                    =   FFFFFFB8     D
SCRIPT$STOP                      ********     X  04
SETPRT$_ACMODE               =   0000000C     D
SETPRT$_INADR                =   00000004     D
SETPRT$_NARGS                =   00000005     D
SETPRT$_PROT                 =   00000010     D
SETPRT$_PRVPRT               =   00000014     D
SETPRT$_RETADR               =   00000008     D
SIZ...                       =   00000001     D
SS$_ACCVIO                       ********     X  04
SS$_BREAK                        ********     X  04
SS$_CONTINUE                     ********     X  04
SS$_NORMAL                   =   00000001     D
SS$_RESIGNAL                     ********     X  04
SS$_ROPRAND                      ********     X  04
SS$_TBIT                         ********     X  04
SYMTAB_HI                        00000185 RG  D  02
SYS$CRETVA                       ********    GX  04
SYS$FAO                          ********     X  04
SYS$FAOL                         ********    GX  04
SYS$SETPRT                       ********     X  04
SYS$UNWIND                       ********    GX  04
TRANS_VECTOR                     00000B2B R   D  04
TRANS_VECTOR_OFFSET          =   00000002     D
T_ACCVIO                         00000183 R   D  03
T_EMESSG1                        ********     X  04
T_MACHCK                         00000162 R   D  03
```

```
T_PREGREAD                      000001D1 R  D  03
T_PREGWRITE                     000001DB R  D  03
T_TRANSL                        00000197 R  D  03
UNMAP_DEBUGGER                  ******** X     04
VRCLRBRK                        00000503 RG D  04
VRCLRBRKX                       0000054D R  D  04
VRDEPOSIT                       000003B2 RG D  04
VRDEPOSITX                      000004AA R  D  04
VREXAMINE                       000000CA RG D  04
VREXAMINEX                      000001BE R  D  04
VRSETBASE                       00000000 RG D  04
VRSETBRK                        000004AD RG D  04
VRSETBRKX                       000004FD R  D  04
VRSETDFLT                       0000001D RG D  04
VRSHOWBASE                      00000009 RG D  04
VRSHOWBRK                       0000057C RG D  04
VRSHOWBRKX                      000005C4 R  D  04
VRSHOWDFLT                      00000060 RG D  04
VRTYPEXAMINE                    000001C6 R  D  04
V_STEP                        = 00000001    D
V_TBIT                        = 00000000    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFF8 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000189 (393.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	000002E8 (744.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000B33 (2867.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...							
$$ARGS	=00000004	177	(3)	124	(2)	177	(3)				
$$N	=00000002	1838	(40)	#-1044	(26)	#-1141	(28)	1143	(28)	#-1148	(28)
				1298	(32)	1629	(39)	#-1802	(40)	#-1826	(40)
				1838	(40)	380	(8)	528	(12)	#-591	(14)
				641	(15)	793	(18)	#-856	(21)	912	(22)
				994	(24)						
$$T1	=00000014	177	(3)	124	(2)	177	(3)				
$$T2	=00000C07	778	(18)	778	(18)						
$ER	=00000001	199	(4)								
$MODULE	00000000-R	199	(4)								
AAP	000002D8-R	292	(6)	274	(6)						
ADR1	FFFFFFD8	1538	(37)	#-1647	(40)	#-1649	(40)	#-1650	(40)	#-1653	(40)
				#-1666	(40)	1670	(40)				
ADR2	FFFFFFB8	1538	(37)	#-1653	(40)	#-1654	(40)	1658	(40)		
AEM	00000254-R	257	(5)	252	(5)						
AFP	000002DB-R	293	(6)	275	(6)						
AKM	0000024D-R	256	(5)	251	(5)						
APC	000002E1-R	295	(6)	277	(6)						
APSL	000002E4-R	296	(6)	278	(6)						
APSLMNE	000001F8-R	246	(5)	763	(17)						
AR0	000002B2-R	280	(6)	262	(6)						
AR1	000002B5-R	281	(6)	263	(6)						
AR10	000002D0-R	290	(6)	272	(6)						
AR11	000002D4-R	291	(6)	273	(6)						
AR2	000002B8-R	282	(6)	264	(6)						
AR3	000002BB-R	283	(6)	265	(6)						
AR4	000002BE-R	284	(6)	266	(6)						
AR5	000002C1-R	285	(6)	267	(6)						
AR6	000002C4-R	286	(6)	268	(6)						
AR7	000002C7-R	287	(6)	269	(6)						
AR8	000002CA-R	288	(6)	270	(6)						
AR9	000002CD-R	289	(6)	271	(6)						
ARG1	FFFFFFE0	1538	(37)	#-1544	(38)	#-1644	(40)	1667	(40)	#-1668	(40)
				#-1669	(40)	#-1671	(40)	#-1672	(40)	1674	(40)
				#-1676	(40)	1694	(40)	1697	(40)		
ARG2	FFFFFFC0	1538	(37)	#-1545	(38)	#-1645	(40)	1655	(40)	#-1656	(40)
				#-1657	(40)	#-1659	(40)	#-1660	(40)	1662	(40)
				#-1664	(40)	1699	(40)	1702	(40)		
ASM	0000025E-R	258	(5)	253	(5)						
ASP	000002DE-R	294	(6)	276	(6)						
AUM	00000269-R	259	(5)	254	(5)						
BASEADDRESS	00000058-R	155	(3)	#-1037	(26)	#-338	(7)	#-380	(8)	#-598	(14)
				#-863	(21)	#-977	(24)	#-985	(24)		
BIT...	=0000000F	126	(2)	120	(2)	121	(2)	126	(2)		
BPTCODE	=00000003	104	(2)	200	(4)						
BPTMAX	=0000000F	106	(2)	#-1048	(26)	#-1095	(27)	#-1136	(28)	#-1184	(29)
				#-1433	(35)	#-1477	(36)	148	(3)	151	(3)
				#-994	(24)						
BPT_LITERAL	00000006-R	200	(4)	1488	(36)						
CALL_CLI	0000064F-R	1300	(32)	#-1283	(32)	#-1390	(34)				

```
CLI$K_BUFSIZ                =00000100     119     (2)     119     (2)
CLI$K_SIZE                   00000444     119     (2)
CLI$L_ADDRESS                00000018     119     (2)   #-1038   (26)   #-338    (7)   #-580   (14)   #-607   (14)
                                                        #-654    (15)   #-658   (15)   #-660   (15)   #-662   (15)
                                                        #-710    (16)   #-760   (17)   #-845   (21)   #-924   (22)
                                                        #-928    (22)   #-931   (22)   #-978   (24)   #-986   (24)
                                                        #-990    (24)
CLI$L_COMMAND                00000004     119     (2)
CLI$L_DATA                   0000001C     119     (2)   #-1034   (26)   #-1389  (34)    887   (22)   #-893   (22)
CLI$L_FLAGS                  00000000     119     (2)     429    (10)     431   (10)    433   (10)    442   (10)
                                                         444    (10)     446   (10)    587   (14)    597   (14)
                                                         605    (14)     606   (14)    611   (14)    613   (14)
                                                         621    (15)     639   (15)    653   (15)    657   (15)
                                                         685    (15)     687   (15)    689   (15)    711   (16)
                                                         715    (16)     736   (17)    739   (17)    759   (17)
                                                         852    (21)     862   (21)    871   (21)    873   (21)
                                                         875    (21)     885   (22)    907   (22)    923   (22)
                                                         927    (22)     935   (22)    974   (24)
CLI$L_LAST                   00000024     119     (2)
CLI$L_NEXT                   00000030     119     (2)   #-651    (15)   #-663   (15)   #-920   (22)   #-932   (22)
CLI$L_PASS                   0000002C     119     (2)
CLI$L_SUBT                   00000028     119     (2)
CLI$L_TEST                   00000020     119     (2)
CLI$Q_BUFQWD                 00000034     119     (2)
CLI$Q_FILE                   00000008     119     (2)    1831    (40)
CLI$Q_SECTION                00000010     119     (2)
CLI$Q_TIME                   0000043C     119     (2)
CLI$T_BUFFER                 0000003C     119     (2)
CLI$V_ADAPTER               =00000018     119     (2)
CLI$V_ADR                   =0000000B     119     (2)
CLI$V_ASCII                 =00000013     119     (2)   #-736    (17)
CLI$V_BREAK                 =0000000A     119     (2)
CLI$V_BRIEF                 =0000001B     119     (2)
CLI$V_BYTE                  =0000000D     119     (2)   #-442    (10)   #-613   (14)   #-875   (21)
CLI$V_CLEAR                 =00000002     119     (2)
CLI$V_DEC                   =00000010     119     (2)   #-431    (10)   #-687   (15)
CLI$V_DEFAULT               =0000000C     119     (2)
CLI$V_DEPOSIT               =00000019     119     (2)
CLI$V_EVENT                 =00000008     119     (2)
CLI$V_EXAM                  =00000005     119     (2)
CLI$V_FLAGS                 =00000009     119     (2)
CLI$V_HEX                   =00000012     119     (2)   #-433    (10)   #-685   (15)
CLI$V_KERNEL                =00000017     119     (2)   #-974    (24)
CLI$V_LOAD                  =00000006     119     (2)
CLI$V_LONG                  =0000000F     119     (2)   #-446    (10)   #-605   (14)   #-871   (21)
CLI$V_NOTNUF                =00000001     119     (2)
CLI$V_OCT                   =00000011     119     (2)   #-429    (10)   #-689   (15)
CLI$V_PREG                  =0000001A     119     (2)   #-597    (14)   #-621   (15)   #-639   (15)   #-657   (15)
                                                        #-715    (16)   #-739   (17)   #-862   (21)   #-885   (22)
                                                        #-907    (22)   #-927   (22)   #-935   (22)
CLI$V_QA                    =00000007     119     (2)
CLI$V_QACKLOOPLOOPS         =0000001C     119     (2)
CLI$V_QAERRORPRINTS         =0000001B     119     (2)
CLI$V_QAMULTIPLEPASS        =0000001F     119     (2)
CLI$V_QASUBTESTLOOPS        =0000001E     119     (2)
CLI$V_QATESTLOOPS           =0000001D     119     (2)
CLI$V_REG                   =00000014     119     (2)   #-587    (14)   #-606   (14)   #-653   (15)   #-711   (16)
```

```
                                                        #-759    (17)   #-852    (21)   #-923    (22)

CLI$V_REQUIRED                  =00000000     119   (2)
CLI$V_RUN                       =00000015     119   (2)
CLI$V_SET                       =00000003     119   (2)
CLI$V_SHOW                      =00000004     119   (2)
CLI$V_VALSEC                    =00000016     119   (2)
CLI$V_WORD                      =0000000E     119   (2)   #-444    (10)   #-611    (14)   #-873    (21)
CMK$SGIPR                        0000098D-R   1720  (40)   #-636    (15)   #-898    (22)
CMK$_                           =00000004     126   (2)
CMK$_ASTEXIT                    =00000000     126   (2)
CMK$_CANTIM                     =0000000B     126   (2)
CMK$_HIBER                      =00000007     126   (2)
CMK$_INITSCB                    =00000008     126   (2)
CMK$_LAST                       =0000000E     126   (2)
CMK$_MMENABLE                   =00000C03     126   (2)
CMK$_RCONSOLE                   =00000001     126   (2)
CMK$_REFRESH                    =00000005     126   (2)
CMK$_SCHDWK                     =00000006     126   (2)
CMK$_SETIMR                     =0000000C     126   (2)
CMK$_SETIPL                     =00000009     126   (2)
CMK$_SETPRT                     =0000000D     126   (2)
CMK$_SGIPR                      =0000000A     126   (2)   #-636    (15)   #-898    (22)
CMK$_TCONSOLE                   =00000002     126   (2)
COND_DEBUG                       000005DE-R   1226  (30)
CONT_FROM_DEBUG                  00000B21-R   1859  (40)     1867   (40)
DATA                             FFFFFFF8     1538  (37)     1557   (38)     1568   (38)
DBG_ADDR_REQ                     0000016D-R   189   (3)   #-1806   (40)   #-1808   (40)   #-1809   (40)     1811   (40)
DBG_ADDR_RTN                     00000175-R   191   (3)     1811   (40)   #-1812   (40)   #-1813   (40)   #-1817   (40)
                                                        #-1818   (40)   #-1822   (40)   #-1828   (40)   #-1838   (40)
                                                        #-1852   (40)
DBG_BREAK                        00000605-R   1266  (32)   #-1232   (30)
DBG_DEF                          000009BF-R   1797  (40)     1830   (40)
DBG_TBIT                         000006D6-R   1369  (34)   #-1237   (30)
DEBUG$GB_FLAGS                   00000000-R   141   (3)    #-1344   (32)     1371   (34)     1375   (34)   #-1393   (34)
DEBUGGER_SIZE                   =00010E00     114   (2)    #-1807   (40)   #-1815   (40)   #-1829   (40)
DEBUG_HI                         00000181-R   194   (3)    #-1813   (40)   #-1821   (40)
DEBUG_LO                         0000017D-R   193   (3)    #-1812   (40)   #-1822   (40)   #-1823   (40)
DFLTSIZE                         00000054-R   152   (3)    #-443    (10)   #-445    (10)   #-447    (10)   #-513    (12)
                                                        #-614    (14)   #-876    (21)
DIAG_FILE_NAME                   00000000-XR                  133    (3)     1848   (40)
DIR...                          =FFFFFFFF     1538  (37)     1538   (37)
DS$AA_BPTADDR                    00000014-R   150   (3)    #-1041   (26)   #-1049   (26)     1093   (27)   #-1137   (28)
                                                        #-1143   (28)   #-1186   (29)   #-1276   (32)   #-1282   (32)
                                                        #-1298   (32)   #-1339   (32)   #-1434   (35)   #-1437   (35)
                                                        #-1478   (36)   #-1482   (36)   #-1487   (36)   #-1493   (36)
                                                        #-987    (24)   #-991    (24)
DS$AB_BPTINST                    00000001-R   147   (3)     1279   (32)     1438   (35)     1481   (36)
DS$CLI                           00000000-XR                 1309   (32)
DS$CVTREG                        00000000-XR                  763   (17)
DS$GL_CLIBASE                    00000000-XR                 1032   (26)     1388   (34)      973   (24)
DS$GL_MEMSIZE                    00000000-XR              #-1816   (40)   #-1820   (40)
DS$GL_RADIX                      00000000-XR              #-430    (10)   #-432    (10)   #-434    (10)   #-502    (12)
                                                        #-692    (15)   #-695    (15)
DS$K_ERROR                      =00000002     121   (2)
DS$K_NORMAL                     =00000001     121   (2)
DS$K_SEVERE                     =00000004     121   (2)
DS$K_SUBSYS                     =00000066     121   (2)      121   (2)
```

```
DS$K_WARNING                    =00000000      121     (2)
DS$M_ABRTFLG                    =00000040      120     (2)
DS$M_BADTIME                    =00100000      120     (2)
DS$M_BATCH                      =00400000      120     (2)
DS$M_BRKCLR                     =00001000      120     (2)
DS$M_BRKPT                      =00000800      120     (2)
DS$M_CHARFLG                    =00000100      120     (2)
DS$M_CMDFLG                     =00000080      120     (2)
DS$M_CTRLC                      =00000001      120     (2)
DS$M_CTRLO                      =00010000      120     (2)
DS$M_DEVFLG                     =00000200      120     (2)
DS$M_DISABLCC                   =01000000      120     (2)
DS$M_DONFLG                     =00002000      120     (2)
DS$M_ERRFLG                     =00000010      120     (2)
DS$M_EXCEPT                     =00080C00      120     (2)
DS$M_EXETST                     =00040000      120     (2)
DS$M_HLTFLG                     =00000008      120     (2)
DS$M_LODFLG                     =00000002      120     (2)
DS$M_MEMMGT                     =00008000      120     (2)
DS$M_OUTPUT                     =00800000      120     (2)
DS$M_RUBFLG                     =00000020      120     (2)
DS$M_SCRIPT                     =00200000      120     (2)
DS$M_SETIMR                     =02000000      120     (2)
DS$M_STRFLG                     =00000004      120     (2)
DS$M_SUBT                       =00004000      120     (2)
DS$M_SYSFLG                     =00000400      120     (2)
DS$M_TIMRON                     =00020000      120     (2)
DS$PRINTF                        00000000-XR                    1044    (26)    1141    (28)    1143    (28)    1148    (28)
                                                                1298    (32)    1386    (34)    1802    (40)    1826    (40)
                                                                1838    (40)    380     (8)     528     (12)    591     (14)
                                                                634     (15)    793     (18)    856     (21)    896     (22)
                                                                994     (24)

DS$V_ABRTFLG                    =00000006      120     (2)
DS$V_BADTIME                    =00000014      120     (2)
DS$V_BATCH                      =00000016      120     (2)
DS$V_BRKCLR                     =0000000C      120     (2)
DS$V_BRKPT                      =0000000B      120     (2)
DS$V_CHARFLG                    =00000008      120     (2)
DS$V_CMDFLG                     =00000007      120     (2)
DS$V_CTRLC                      =00000000      120     (2)
DS$V_CTRLO                      =00000010      120     (2)
DS$V_DEVFLG                     =00000009      120     (2)
DS$V_DISABLCC                   =00000018      120     (2)
DS$V_DONFLG                     =0000000D      120     (2)
DS$V_ERRFLG                     =00000004      120     (2)
DS$V_EXCEPT                     =00000013      120     (2)
DS$V_EXETST                     =00000012      120     (2)
DS$V_HLTFLG                     =00000003      120     (2)
DS$V_LODFLG                     =00000001      120     (2)
DS$V_MEMMGT                     =0000000F      120     (2)
DS$V_OUTPUT                     =00000017      120     (2)
DS$V_RUBFLG                     =00000005      120     (2)
DS$V_SCRIPT                     =00000015      120     (2)
DS$V_SETIMR                     =00000019      120     (2)
DS$V_STRFLG                     =00000002      120     (2)
DS$V_SUBT                       =0000000E      120     (2)
DS$V_SYSFLG                     =0000000A      120     (2)
```

```
DS$V_TIMRON                     =00000011      120     (2)
DS$_ARITH                       =006600D0      121     (2)
DS$_ASBE                        =00660118      121     (2)
DS$_BADLINK                     =006600F0      121     (2)
DS$_BADTYPE                     =006600E8      121     (2)
DS$_BIIC                        =00660120      121     (2)
DS$_CHME                        =006600A8      121     (2)
DS$_CHMK                        =006600E0      121     (2)
DS$_DEVNAME                     =00660108      121     (2)
DS$_ERROR                       =00660002      121     (2)
DS$_FHWE                        =00660068      121     (2)
DS$_FRAGBUF                     =00660080      121     (2)
DS$_ICBUSY                      =006600C8      121     (2)
DS$_ICERR                       =006600C0      121     (2)
DS$_IHWE                        =00660060      121     (2)
DS$_ILLCHAR                     =00660018      121     (2)
DS$_ILLPAGCNT                   =00660078      121     (2)
DS$_ILLUNIT                     =00660100      121     (2)
DS$_INSFMEM                     =00660050      121     (2)
DS$_IPL2HI                      =006600B8      121     (2)
DS$_IVADDR                      =00660040      121     (2)
DS$_IVVECT                      =00660038      121     (2)
DS$_KRNLSTK                     =00660090      121     (2)
DS$_LOGIC                       =00660070      121     (2)
DS$_MCHK                        =00660088      121     (2)     #-1605    (39)
DS$_MMOFF                       =00660058      121     (2)
DS$_NEEDUNIT                    =006600F8      121     (2)
DS$_NODE                        =00660128      121     (2)
DS$_NOPCS                       =00660110      121     (2)
DS$_NORMAL                      =00660001      121     (2)
DS$_NOSUPPORT                   =006600B1      121     (2)
DS$_NOTDON                      =00660030      121     (2)
DS$_NOTIMP                      =006600B0      121     (2)       121     (2)
DS$_NULLSTR                     =00660010      121     (2)
DS$_OVERFLOW                    =00660008      121     (2)
DS$_POWER                       =00660098      121     (2)
DS$_PROGERR                     =00660020      121     (2)
DS$_SEVERE                      =00660004      121     (2)
DS$_TRANSL                      =006600A0      121     (2)     #-1610    (39)
DS$_TRUNCATE                    =00660028      121     (2)
DS$_UNEXPINT                    =006600D8      121     (2)
DS$_VASFULL                     =00660048      121     (2)
DS$_WARNING                     =00660000      121     (2)       121     (2)
DSA$GL_FLAGS                     0000FE00                        1800    (40)      1804    (40)    #-1840    (40)      1844    (40)
                                                                 632    (15)       894    (22)
DSA$V_DEBUG                     =0000001A                      #-1799    (40)    #-1839    (40)
DSA$V_USER                      =0000001C                      #-1803    (40)    #-1843    (40)    #-632     (15)    #-894     (22)
DSP$REMOVEBREAK                  00000553-R     1090    (27)
DSR$COMPLETION                   00000000-XR                     1833    (40)
DSV$DEBUG                        000009CB-R     1798    (40)
DSV$DEBUG_EXIT                   00000B2A-R     1863    (40)   #-1827    (40)    #-1836    (40)
DSX$LOAD                         00000000-XR                     1832    (40)
DSX$PRINTI                       00000000-XR                     1629    (39)      1684    (40)     641    (15)     912    (22)
EXAMBUFFER                       00000083-R      168    (3)      167    (3)       707    (16)     790    (18)
EXAMQWDBUF                       0000007B-R      166    (3)      177    (3)     #-706    (16)    #-707    (16)    #-716    (16)
                                                               #-717    (16)    #-718    (16)    #-723    (16)    #-724    (16)
                                                               #-725    (16)    #-727    (16)    #-728    (16)    #-729    (16)
```

```
                                                #-751    (17)   #-752    (17)   #-753    (17)    778     (18)
                                                #-782    (18)   #-783    (18)   #-789    (18)  #-790     (18)
                                                #-793    (18)
EXAMSIZE                =00000020       107     (2)      167    (3)      169    (3)    #-706    (16)  #-789     (18)
FAOARG                   00000107-R     179     (3)      177    (3)    #-709    (16)  #-713    (16)  #-720     (16)
                                                         #-749    (17)
FAOL$_CTRSTR            =00000004       177     (3)
FAOL$_NARGS             =00000004       177     (3)
FAOL$_OUTBUF            =0000000C       177     (3)
FAOL$_OUTLEN            =00000008       177     (3)
FAOL$_PRMLST            =00000010       177     (3)
FAOLST                   000000F3-R     173     (3)      722    (16)     750    (17)
FETCH_STORE              000007C3-R     1540    (38)    1281    (32)    1381    (34)    1440    (35)    1484    (36)
                                                        1490    (36)     627    (15)     889    (22)
FETCH_STOR_PREG          0000099C-R     1764    (40)    1725    (40)
FIND_BPT                 000005C7-R     1183    (29)  #-1039    (26)  #-1285    (32)  #-979    (24)  #-983     (24)
FMTBPT                   00000085-R     218     (4)     1298    (32)
FMTBPTADR                000000A7-R     222     (4)     1143    (28)
FMTBPTLST                000000B2-R     224     (4)     1141    (28)
FMTBPTNONE               000000C9-R     226     (4)     1148    (28)
FMTBPTNOTSET             000000DE-R     228     (4)     1044    (26)
FMTDIRECTIVE             00000070-R     162     (3)      177    (3)    #-684    (15)  #-699    (15)  #-712     (16)
                                                       #-719    (16)   #-747    (17)  #-748    (17)
FMTEXAM                  0000012B-R     234     (4)      793    (18)
FMTEXAM_ASCII            00000118-R     232     (4)      778    (18)
FMTINVLDREG              000000F9-R     230     (4)      591    (14)     856    (21)
FMTNOBPTSLEFT            00000139-R     236     (4)      994    (24)
FMTSTEP                  0000009C-R     220     (4)     1385    (34)
FMT_BADPREG              000001E4-R     245     (5)      641    (15)     912    (22)
FMT_BASE_OUT             00000007-R     202     (4)      380    (8)
FMT_BYTE                 00000080-R     216     (4)      522    (12)
FMT_DBG_LOAD             00000121-R     187     (3)     1838    (40)
FMT_DBG_MSG              0000010B-R     186     (3)     1802    (40)
FMT_DEC                  00000064-R     208     (4)      509    (12)
FMT_DFLT_OUT             0000002D-R     204     (4)      528    (12)
FMT_EXCEPTION            000001B0-R     242     (5)     1629    (39)    1683    (40)
FMT_HEX                  00000058-R     206     (4)      505    (12)
FMT_LONG                 00000072-R     212     (4)      516    (12)
FMT_NO_ROOM              0000013A-R     188     (3)     1826    (40)
FMT_OCT                  0000006C-R     210     (4)      511    (12)
FMT_WORD                 0000007B-R     214     (4)      520    (12)
FS_COPY                  0000081A-R     1582    (39)  #-1558    (38)  #-1569    (38)
FS_HANDLR                0000083F-R     1601    (39)    1547    (38)    1765    (40)
FS_RESET                 00000964-R     1693    (40)  #-1559    (38)  #-1570    (38)  #-1632    (39)  #-1686    (40)
FS_SETUP                 000008CE-R     1642    (40)  #-1554    (38)  #-1566    (38)
HTAB                    =00000009       103     (2)
IMAGE_HEADER             00000000-XR                    1849    (40)
INITBASE                =00000000       109     (2)      156    (3)
INITSIZE                =00000004       110     (2)      153    (3)
LOCAL                    FFFFFFB8       1538    (37)    1543    (38)
MAP_DEBUGGER             00000000-XR                    1824    (40)
MODELST                  00000239-R     250     (5)      763    (17)
M_STEP                  =00000002       146     (3)   #-1393    (34)
M_TBIT                  =00000001       144     (3)   #-1393    (34)
NEW_AP_FP                0000005C-R     158     (3)   #-1318    (32)  #-1340    (32)
NEW_PC_PSL               00000068-R     160     (3)   #-1321    (32)  #-1342    (32)
NEW_SP                   00000064-R     159     (3)   #-1319    (32)  #-1341    (32)
```

```
PRT$C_UR                        00000000-XR                     #-1553  (38)
PRT$C_UW                        00000000-XR                     #-1565  (38)
PSL$C_KERNEL                    =00000000                       #-1551  (38)  #-1563  (38)
PSL$M_TBIT                      =00000010                       #-1320  (32)  #-1343  (32)  #-1395  (34)  #-1399  (34)
PSL$M_TP                        =40000000                       #-1320  (32)  #-1399  (34)
PSL$V_IS                        =0000001A       126     (2)     #-636   (15)  #-898   (22)
PSLBUFFER                       000000A3-R      170     (3)     #-735   (17)    763   (17)  #-793   (18)
PSLBUFSZ                        =00000050       108     (2)       171   (3)   #-763   (17)
PSLREG                          =00000010       105     (2)     #-607   (14)  #-654   (15)  #-760   (17)  #-924   (22)
RBREAK_IN                       00000775-R      1474    (36)    #-1053  (26)  #-1104  (27)  #-1373  (34)  #-671   (15)
                                                                #-903   (22)  #-997   (24)
RBREAK_OUT                      0000074A-R      1431    (35)    #-1031  (26)  #-1092  (27)  #-1337  (32)  #-1392  (34)
                                                                #-577   (14)  #-883   (22)  #-972   (24)
REGNAMLST                       0000026E-R      261     (6)     #-713   (16)
SAVABS...                       =FFFFFFB8       1538    (37)
SCRIPT$STOP                     00000000-XR                     1308    (32)
SETPRT$_ACMODE                  =0000000C       124     (2)
SETPRT$_INADR                   =00000004       124     (2)     #-1659  (40)  #-1671  (40)
SETPRT$_NARGS                   =00000005       124     (2)     #-1660  (40)  #-1672  (40)
SETPRT$_PROT                    =00000010       124     (2)     #-1544  (38)  #-1545  (38)  #-1644  (40)  #-1645  (40)
                                                                  1655  (40)  #-1664  (40)    1667  (40)  #-1676  (40)
                                                                  1694  (40)    1699  (40)
SETPRT$_PRVPRT                  =00000014       124     (2)     #-1656  (40)  #-1668  (40)
SETPRT$_RETADR                  =00000008       124     (2)     #-1657  (40)  #-1669  (40)
SIZ...                          =00000001       120     (2)       120   (2)
SS$_ACCVIO                      00000000-XR                     #-161?  (39)
SS$_BREAK                       00000000-XR                     #-123J  (30)
SS$_CONTINUE                    00000000-XR                     #-1325  (32)  #-1401  (34)
SS$_NORMAL                      =00000001       112     (2)     #-1191  (29)  #-1571  (38)
SS$_RESIGNAL                    00000000-XR                     #-1228  (30)  #-1288  (32)  #-1370  (34)  #-1626  (39)
SS$_ROPRAND                     00000000-XR                     #-1621  (39)
SS$_TBIT                        00000000-XR                     #-1235  (30)
SYMTAB_HI                       00000185-R      195     (3)     #-1823  (40)
SYS$CRETVA                      00000000-XR                       1811  (40)
SYS$FAO                         00000000-XR                       778   (18)
SYS$FAOL                        00000000-XR                       722   (16)    750   (17)
SYS$SETPRT                      00000000-XR                       1662  (40)    1674  (40)    1697  (40)    1702  (40)
SYS$UNWIND                      00000000-XR                       1633  (39)
TRANS_VECTOR                    00000B28-R      1865    (40)      1851  (40)
TRANS_VECTOR_OFFSET             =00000002       1793    (40)    #-1854  (40)
T_ACCVIO                        00000183-R      240     (5)       1618  (39)    1682  (40)
T_EMESSG1                       00000000-XR                       633   (15)    895   (22)
T_MACHCK                        00000162-R      239     (5)       1607  (39)
T_PREGREAD                      000001D1-R      243     (5)       640   (15)    911   (22)
T_PREGWRITE                     000001DB-R      244     (5)       909   (22)
T_TRANSL                        00000197-R      241     (5)       1613  (39)
UNMAP_DEBUGGER                  00000000-XR                       1835  (40)
VRCLRBRK                        00000503-R      1029    (26)
VRCLRBRKX                       00000540-R      1052    (26)    #-1042  (26)  #-1045  (26)
VRDEPOSIT                       000003B2-R      841     (21)
VRDEPOSITX                      000004AA-R      939     (22)    #-857   (21)  #-897   (22)  #-907   (22)  #-913   (22)
                                                                #-936   (22)
VREXAMINE                       000000CA-R      575     (14)
VREXAMINEX                      000001BE-R      670     (15)    #-592   (14)  #-635   (15)  #-642   (15)  #-652   (15)
                                                                #-655   (15)  #-659   (15)
VRSETBASE                       00000000-R      337     (7)
VRSETBRK                        000004AD-R      970     (24)
```

VRSETBRKX	000004FD-R	996	(24)	#-980	(24)	#-988	(24)	#-992	(24)
VRSETDFLT	0000001D-R	422	(10)						
VRSHOWBASE	00000009-R	378	(8)						
VRSHOWBRK	0000057C-R	1132	(28)						
VRSHOWBRKX	000005C4-R	1150	(28)	#-1146	(28)				
VRSHOWDFLT	00000060-R	493	(12)						
VRTYPEXAMINE	000001C6-R	681	(15)	#-643	(15)	#-937	(22)		
V_STEP	=00000001	145	(3)	#-1375	(34)	146	(3)		
V_TBIT	=00000000	143	(3)	#-1344	(32)	#-1371	(34)	144	(3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CRETVA_S	1	1810 (40)	1810 (40)
$D1_PRINT_S	1	380 (8)	1044 (26) 1141 (28) 1143 (28) 1148 (28) 1298 (32)
			1629 (39) 1802 (40) 1826 (40) 1838 (40) 380 (8)
			528 (12) 591 (14) 641 (15) 793 (18) 856 (21)
			912 (22) 994 (24)
$DEF	1	126 (2)	
$DEFINI	1	122 (2)	122 (2) 123 (2) 125 (2)
$DS_CVTREG_S	1	762 (17)	762 (17)
$DS_DSADEF	5	125 (2)	125 (2)
$DS_DSDEF	2	121 (2)	121 (2)
$DS_PRINTF_S	1	379 (8)	1044 (26) 1141 (28) 1143 (28) 1148 (28) 1298 (32)
			1802 (40) 1826 (40) 1837 (40) 379 (8) 527 (12)
			591 (14) 791 (18) 856 (21) 994 (24)
$EQU	1	126 (2)	121 (2) 126 (2)
$EQULS1	1	126 (2)	121 (2) 126 (2)
$EQULST	1	121 (2)	121 (2) 126 (2)
$FAOL	1	174 (3)	174 (3)
$FAOLDEF	1	177 (3)	177 (3)
$FAOL_G	1	722 (16)	722 (16) 750 (17)
$FAO_S	2	776 (18)	776 (18)
$GBLINI	2	120 (2)	120 (2) 121 (2) 126 (2)
$OFFDEF	1	124 (2)	124 (2) 177 (3)
$OFFSET	2	1532 (37)	1532 (37)
$OFFST1	1	1538 (37)	1538 (37)
$PRDEF	4	123 (2)	123 (2)
$PRINTI_S	1	641 (15)	1629 (39) 641 (15) 912 (22)
$PSLDEF	2	122 (2)	122 (2)
$PUSHADR	1	763 (17)	1633 (39) 1811 (40) 763 (17) 778 (18)
$SETPRTDEF	1	124 (2)	124 (2)
$UNWIND_S	1	1633 (39)	1633 (39)
$VIELD	1	120 (2)	120 (2)
$VIELD1	1	126 (2)	120 (2)
CLIDEF	3	119 (2)	119 (2)
CMK	1	636 (15)	636 (15) 898 (22)
CMKDEF	1	126 (2)	126 (2)
DSFDEF	3	120 (2)	120 (2)
MODNAM	1	199 (4)	199 (4)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.24
Command processing	135	00:00:00.71	00:00:01.52
Pass 1	956	00:00:15.07	00:00:18.95
Symbol table sort	0	00:00:00.88	00:00:01.03
Pass 2	413	00:00:04.99	00:00:06.56
Symbol table output	1	00:00:00.25	00:00:00.26

Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	78	00:00:01.77	00:00:02.12
Assembler run totals	1624	00:00:23.80	00:00:30.71

The working set limit was 1000 pages.
115562 bytes (226 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 529 non-local and 110 local symbols.
1869 source lines were read in Pass 1, producing 0 object records in Pass 2.
77 pages of virtual memory were used to define 32 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	5
DRB1:[DS.WORK]DS.MLB;218	6
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	29

587 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DEBUG/UPDA=(DEBUG.UPD,DEBUG.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

Table of contents

(1) 69 ALLOCATE DEVICE
(1) 202 DEALLOCATE DEVICE
(1) 283 LOCK I/O DATA BASE AND SEARCH FOR DEVICE
(1) 300 DECREMENT REFERENCE COUNT, CLEAR OWNERSHIP, AND CALL DRIVER
(1) 324.2 DSX$MOUNT - Mount a device
(1) 324.39 DSX$DISMOUNT - Dismount a device

```
                        0000     .1          .TITLE  DEVALC  *** DEVALC device allocation routines
                        0000     .2          .IDENT  /05-03/
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000     .1 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980, 1984                              *
-1                      0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 9-SEP-76
                        0000     27 ;
                        0000     28 ; MODIFICATION HISTORY:
                        0000     29 ;
                        0000     .1 ;                N. HOWGATE      17-FEB-78       VERSION 02 (ESSAA-4.00).
                        0000     .2 ;       01       DIAGNOSTIC SUPERVISOR INTEGRATION
                        0000     .3 ;                Dave Butenhof   13-may-1980, Version 5.4
                        0000     .4 ;       02       Modify VMS V2.0 source to supervisor environment.
                        0000     .5 ;
                        0000     30 ;       V02      LMK0001         LEN KAWELL                20-JAN-1980
                        0000     31 ;                ADDED NONSHAREABLE DEVICE ALLOCATION PRIVILEGE CHECK.
                        0000     32 ;
                        0000     .1 ;       [03]     Richard Brown   12-July-84      Version 7.0
                        0000     .2 ;                Added DSX$MOUNT and DSX$DISMOUNT
                        0000     .3 ;
                        0000     .4 ;
                        0000     33 ; SYSTEM SERVICES FOR DEVICE ALLOCATION
                        0000     34 ;
                        0000     35 ;       ALLOCATE DEVICE
                        0000     36 ;       DEALLOCATE DEVICE
                        0000     37 ;
                        0000     38 ; MACRO LIBRARY CALLS
                        0000     39 ;
                        0000     40
                        0000     .1         $ALLOCDEF                       ; $ALLOC CALL OFFSETS
                        0000     .2         $DALLOCDEF                      ;$DALLOC CALL OFFSETS
                        0000     41         $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     42         $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000     43         $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
                        0000     44         $PCBDEF                         ;DEFINE PCB OFFSETS
                        0000     45         $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     46         $PRVDEF                         ;DEFINE PRIVILEGE BITS
```

```
                                0000     47          $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                                0000     48          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                                0000     49          $UCBDEF                         ;DEFINE UCB OFFSETS
                                0000     .1
                                0000     .2          $DS_DSADEF                      ; Define VDS flags              [03]
-1                              0000     51 ;
                                0000     52 ; LOCAL SYMBOLS
                                0000     53 ;
                                0000     54 ; ARGUMENT LIST OFFSET DEFINITIONS FOR ALLOCATE DEVICE
                                0000     55 ;
                                0000     56
                    00000004    0000     57 DEVNAM=4                                ;ADDRESS OF DEVICE NAME STRING DESCRIPTOR
                    00000008    0000     58 PHYLEN=8                                ;ADDRESS TO STORE LENGTH OF PHYSICAL NAME
                    0000000C    0000     59 PHYBUF=12                               ;ADDRESS OF PHYSICAL NAME BUFFER DESCRIPTOR
                    00000010    0000     60 ALACMODE=16                             ;ACCESS MODE
                                0000     61
                                0000     62 ;
                                0000     63 ; ARGUMENT LIST OFFSET DEFINITIONS FOR DEALLOCATE DEVICE
                                0000     64 ;
                                0000     65
                    00000004    0000     66 DEVNAM=4                                ;ADDRESS OF DEVICE NAME STRING DESCRIPTOR
                    00000008    0000     67 DLACMODE=8                              ;ACCESS MODE
```

```
                        .SBTTL  ALLOCATE DEVICE
                ;+
                ; EXE$ALLOC - ALLOCATE DEVICE
                ;
                ; THIS SERVICE PROVIDES THE CAPABILITY TO RESERVE A DEVICE FOR EXCLUSIVE
                ; USE.
                ;
                ; INPUTS:
                ;
                ;       DEVNAM(AP) = ADDRESS OF DEVICE NAME STRING DESCRIPTOR.
                ;       PHYLEN(AP) = ADDRESS TO STORE LENGTH OF PHYSICAL NAME.
                ;       PHYBUF(AP) = ADDRESS OF PHYSICAL NAME BUFFER DESCRIPTOR.
                ;       ALACMODE(AP) = ACCESS MODE DEVICE IS TO BE ALLOCATED TO.
                ;
                ;       R4 = CURRENT PROCESS PCB ADDRESS.
                ;
                ; OUTPUTS:
                ;
                ;       R0 LOW BIT CLEAR INDICATES FAILURE TO ALLOCATE DEVICE.
                ;
                ;               R0 = SS$_ACCVIO - DEVICE NAME STRING, DEVICE NAME STRING
                ;                       DESCRIPTOR, OR PHYSICAL NAME BUFFER DESCRIPTOR
                ;                       CANNOT BE READ BY CALLING ACCESS MODE, OR PHYSICAL
                ;                       NAME BUFFER CANNOT BE WRITTEN BY CALLING ACCESS
                ;                       MODE.
                ;
                ;               R0 = SS$_DEVALLOC - DEVICE ALREADY ALLOCATED TO ANOTHER
                ;                       PROCESS.
                ;
                ;               R0 = SS$_DEVMOUNT - DEVICE CURRENTLY MOUNTED AND CANNOT
                ;                       BE ALLOCATED.
                ;
                ;               R0 = SS$_IVDEVNAM - DEVICE NAME STRING CONTAINS INVALID
                ;                       CHARACTERS OR NO DEVICE NAME STRING DESCRIPTOR
                ;                       SPECIFIED.
                ;
                ;               R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
                ;                       DEVICE NAME STRING SPECIFIED.
                ;
                ;               R0 = SS$_NONLOCAL - SPECIFIED DEVICE EXISTS ON A REMOTE
                ;                       SYSTEM.
                ;
                ;               R0 = SS$_NOPRIV - DEVICE CURRENTLY SPOOLED AND PROCESS DOES
                ;                       NOT HAVE PRIVILEGE TO ALLOCATE.
                ;
                ;               R0 = SS$_NOSUCHDEV - SPECIFIED DEVICE DOES NOT EXIST ON
                ;                       HOST SYSTEM OR NO ALLOCATABLE UNIT CAN BE FOUND.
                ;
                ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                ;
                ;               R0 = SS$_DEVALRALLOC - DEVICE IS ALREADY ALLOCATED TO CALLING
                ;                       PROCESS.
                ;
                ;               R0 = SS$_BUFFEROVF - NORMAL COMPLETION, PHYSICAL NAME OVER-
                ;                       FLOWED PHYSICAL NAME BUFFER.
                ;
                ;               R0 = SS$_NORMAL - NORMAL COMPLETION, PHYSICAL NAME TRANSFERED
```

```
                                0000   126 ;                           TO PHYSICAL NAME BUFFER.
                                0000   127 ;-
                                0000   128
                                00000000   .1          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                                0000   .2          MODNAM  SYSDEVALC
                              007C 000A   .3          .ENTRY  EXE$ALLOC,^M<R2,R3,R4,R5,R6>
                00000000'EF     9E 000C   .4          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
                           54           0012
-2              55  0144 8F     3C 0013   131         MOVZWL  #SS$_IVDEVNAM,R5        ;SET INVALID DEVICE NAME STATUS
                       00ED     30 0018   132         BSBW    SEARCHDEV               ;LOCK I/O DATA BASE AND GET DEVICE NAME
                          3F    13 001B   133         BEQL    27$                     ;IF EQL NO DEVICE NAME SPECIFIED
                        FFE0'   30 001D   134         BSBW    IOC$SEARCHGEN           ;SEARCH FOR GENERIC DEVICE
                     55    50   D0 0020   135         MOVL    R0,R5                   ;SAVE SEARCH STATUS
                        36 55   E9 0023   136         BLBC    R5,27$                  ;IF LBC SEARCH FAILURE
                     56    51   D0 0026   137         MOVL    R1,R6                   ;SAVE DEVICE UCB ADDRESS
                                0029   138 ;
                                0029   139 ; RETURN PHYSICAL DEVICE NAME
                                0029   140 ;
                  51    0C AC   D0 0029   141         MOVL    PHYBUF(AP),R1           ;GET ADDRESS OF NAME BUFFER DESCRIPTOR
                          1C    13 002D   142         BEQL    20$                     ;IF EQL NONE
-1                   50   61   3C 002F   144         MOVZWL  (R1),R0                 ;GET LENGTH OF NAME BUFFER
                          17    13 0032   145         BEQL    20$                     ;IF EQL ZERO LENGTH BUFFER
                  51    04 A1   D0 0034   146         MOVL    4(R1),R1                ;GET ADDRESS OF NAME BUFFER
                  53    08 AC   D0 0038   .1          MOVL    PHYLEN(AP),R3           ;GET ADDRESS TO STORE NAME LENGTH
                          00    13 003C   .2          BEQL    10$                     ;IF EQL NONE
-4                    FFBF'   30 003E   151 10$:      BSBW    IOC$CVT_DEVNAM          ;CONVERT DEVICE NAME AND UNIT
                     55   50   D0 0041   152         MOVL    R0,R5                   ;SAVE COMPLETION STATUS
                          53    D5 0044   153         TSTL    R3                      ;ADDRESS TO STORE LENGTH SPECIFIED?
                          03    13 0046   154         BEQL    20$                     ;IF EQL NONE SPECIFIED
                     63   51   B0 0048   155         MOVW    R1,(R3)                 ;INSERT LENGTH OF CONVERTED STRING
                                004B   156 ;
                                004B   157 ; CHECK IF PROCESS ALREADY ALLOCATED DEVICE OR CAN ALLOCATE THE DEVICE
                                004B   158 ;
                     50    28 A6   D0 004B   159 20$:      MOVL    UCB$L_PID(R6),R0        ;GET PROCESS ID OF OWNER
                          0D    13 004F   160         BEQL    30$                     ;IF EQL DEVICE NOT OWNED
                     60 A4   50   D1 0051   161         CMPL    R0,PCB$L_PID(R4)        ;OWNED BY CURRENT PROCESS?
                          1D    13 0055   162         BEQL    60$                     ;IF EQL YES
                  55    0840 8F     3C 0057   163 25$:      MOVZWL  #SS$_DEVALLOC,R5        ;SET DEVICE ALREADY ALLOCATED
                          3F    11 005C   164 27$:      BRB     80$                     ;
                                005E   165
                02 34 A6    06    E1 005E   166 30$:      BBC     #DEV$V_SPL,UCB$L_DEVCHAR(R6),40$ ;IF CLR, DEVICE NOT SPOOLED
-1                            05    11 0063   168         BRB     47$                     ;
                        50 A6   B5 0065   169 40$:      TSTW    UCB$W_REFC(R6)          ;ANY CHANNELS ASSIGNED TO DEVICE?
                          ED    12 0068   170         BNEQ    25$                     ;IF NEQ YES
                       00AB     31 006A   .1 47$:      BRW     NOPRIV                  ;ELSE NO PRIVILEGE
                          2E    11 006D   .2 50$:      BRB     80$                     ;
-11                  55   0C   3C 006F   182 55$:      MOVZWL  #SS$_ACCVIO,R5          ;SET ACCESS VIOLATION STATUS
                          29    11 0072   183         BRB     80$                     ;
                                0074   184
                     55   01   B1 0074   185 60$:      CMPW    #SS$_NORMAL,R5          ;NORMAL COMPLETION STATUS?
                          0A    12 0077   186         BNEQ    70$                     ;IF NEQ NO
                05 34 A6    17    E1 0079   187         BBC     #DEV$V_ALL,UCB$L_DEVCHAR(R6),70$ ;IF CLR, DEVICE NOT ALLOCATED
                 55    0641 8F     3C 007E   188         MOVZWL  #SS$_DEVALRALLOC,R5     ;SET DEVICE ALREADY ALLOCATED STATUS
                                0083   189 ;
                                0083   190 ; ALLOCATE THE DEVICE BY SETTING OWNER PID, SETTING ALLOCATED CHARACTERISTIC,
                                0083   191 ; INCREMENTING THE REFERENCE COUNT, AND SETTING THE ALLOCATOR'S ACCESS MODE
                                0083   192 ;
```

```
                28 A6    60 A4   D0  0083   193 70$:    MOVL    PCB$L_PID(R4),UCB$L_PID(R6) ;SET OWNER PROCESS ID
                10 34 A6   17   E2  0088   194          BBSS    #DEV$V_ALL,UCB$L_DEVCHAR(R6),80$ ;IF SET, DEVICE ALREADY ALLOCATED
                      50 A6    B6  008D   195          INCW    UCB$W_REFC(R6)              ;INCREMENT REFERENCE COUNT
                   02   00    EF  0090   196          EXTZV   #0,#2,ALACMODE(AP),R0       ;GET SPECIFIED ACCESS MODE
                50   10 AC          0093
                      FF67'    30  0096   197          BSBW    EXE$MAXACMODE               ;MAXIMIZE ACCESS MODE
                53 A6    50    90  0099   198          MOVB    R0,UCB$B_AMOD(R6)           ;SET ALLOCATING ACCESS MODE
                   50    55    D0  009D   199 80$:     MOVL    R5,R0                       ;SET FINAL STATUS
                      FF5D'    31  00A0   200          BRW     IOC$UNLOCK                  ;
```

```
                                00A3   202          .SBTTL  DEALLOCATE DEVICE
                                00A3   203 ;+
                                00A3   204 ; EXE$DALLOC - DEALLOCATE DEVICE
                                00A3   205 ;
                                00A3   206 ; THIS SERVICE PROVIDES THE CAPABILITY TO RELINQUISH EXCLUSIVE USE OF A
                                00A3   207 ; DEVICE.
                                00A3   208 ;
                                00A3   209 ; INPUTS:
                                00A3   210 ;
                                00A3   211 ;       DEVNAM(AP) = ADDRESS OF DEVICE NAME STRING DESCRIPTOR. ZERO
                                00A3   212 ;                    IMPLIES ALL.
                                00A3   213 ;       DLACMODE(AP) = ACCESS MODE FOR DEALLOCATION.
                                00A3   214 ;
                                00A3   215 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                00A3   216 ;
                                00A3   217 ; OUTPUTS:
                                00A3   218 ;
                                00A3   219 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO DEALLOCATE DEVICE.
                                00A3   220 ;
                                00A3   221 ;               R0 = SS$_ACCVIO - DEVICE NAME STRING OR DEVICE NAME STRING
                                00A3   222 ;                    DESCRIPTOR CANNOT BE READ BY CALLING ACCESS MODE.
                                00A3   223 ;
                                00A3   224 ;               R0 = SS$_DEVASSIGN - DEVICE CANNOT BE DEALLOCATED BECAUSE
                                00A3   225 ;                    PROCESS STILL HAS CHANNELS ASSIGNED.
                                00A3   226 ;
                                00A3   227 ;               R0 = SS$_DEVNOTALLOC - DEVICE NOT ALLOCATED OR NOT ALLOCATED
                                00A3   228 ;                    TO PROCESS.
                                00A3   229 ;
                                00A3   230 ;             R0 = SS$_IVDEVNAM - DEVICE NAME STRING CONTAINS INVALID
                                00A3   231 ;                  CHARACTERS.
                                00A3   232 ;
                                00A3   233 ;             R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
                                00A3   234 ;                  DEVICE NAME STRING SPECIFIED.
                                00A3   235 ;
                                00A3   236 ;             R0 = SS$_NOPRIV - CALLING ACCESS MODE DOES NOT HAVE PRIVILEGE
                                00A3   237 ;                  TO DEALLOCATE DEVICE.
                                00A3   238 ;
                                00A3   239 ;             R0 = SS$_NOSUCHDEV - SPECIFIED DEVICE DOES NOT EXIST ON
                                00A3   240 ;                  HOST SYSTEM.
                                00A3   241 ;
                                00A3   242 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                00A3   243 ;
                                00A3   244 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                00A3   245 ;-
                                00A3   246
                          003C  00A3   247          .ENTRY  EXE$DALLOC,^M<R2,R3,R4,R5>
      00000000'EF     9E        00A5    .1          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
                      54        00AB
-3                    5A   10   00AC   251          BSBB    SEARCHDEV               ;LOCK I/O DATA BASE AND SEARCH FOR DEVICE
                      1B   13   00AE   252          BEQL    10$                     ;IF EQL NO DEVICE NAME SPECIFIED
                  FF4D'    30   00B0   253          BSBW    IOC$SEARCHDEV           ;SEARCH FOR PHYSICAL DEVICE
                   50 50   E9   00B3   254          BLBC    R0,50$                  ;IF LBC SEARCH FAILURE
            60 A4  28 A1   D1   00B6   255          CMPL    UCB$L_PID(R1),PCB$L_PID(R4) ;DEVICE ALLOCATED TO CURRENT PROCESS?
                      44   12   00BB   256          BNEQ    40$                     ;IF NEQ NO
-4                 50 A1   01   B1   00BD   261     CMPW    #1,UCB$W_REFC(R1)       ;REFERENCE COUNT ONE?
                      43   12   00C1   262          BNEQ    50$                     ;IF NEQ NO
            00000123'EF    16   00C3   263          JSB     DECREF                  ;DECREMENT REFERENCE COUNT AND CALL DRIVER
```

```
                52   11  00C9    264          BRB     NORMAL                          ;
      00000000'EF   9E  00CB    265 10$:     MOVAB   L^IOC$GL_DEVLIST-DDB$L_LINK,R3 ;GET ADDRESS OF I/O DATABASE LISTHEAD
                53         00D1
           53   63   D0  00D2    266 20$:     MOVL    DDB$L_LINK(R3),R3               ;GET ADDRESS OF NEXT DDB
                46   13  00D5    267          BEQL    NORMAL                          ;IF EQL END OF LIST
         51    D8 A3   9E  00D7    268          MOVAB   DDB$L_UCB-UCB$L_LINK(R3),R1 ;GET ADDRESS OF NEXT UCB ADDRESS
         51    2C A1   D0  00DB    269 30$:     MOVL    UCB$L_LINK(R1),R1               ;GET ADDRESS OF NEXT UCB
                F1   13  00DF    270          BEQL    20$                             ;IF EQL END OF LIST
      60 A4    28 A1   D1  00E1    271          CMPL    UCB$L_PID(R1),PCB$L_PID(R4) ;DEVICE ALLOCATED TO CURRENT PROCESS?
                F3   12  00E6    272          BNEQ    30$                             ;IF NEQ NO
         53 A1   55   91  00E8    273          CMPB    R5,UCB$B_AMOD(R1)               ;CAN ACCESS MODE DEALLOCATE DEVICE?
                ED   14  00EC    274          BGTR    30$                             ;IF GTR NO
      E8 34 A1   17   E1  00EE    275          BBC     #DEV$V_ALL,UCB$L_DEVCHAR(R1),30$ ;IF CLR, DEVICE NOT ALLOCATED
         50 A1   01   B1  00F3    276          CMPW    #1,UCB$W_REFC(R1)               ;REFERENCE COUNT ONE?
                E2   12  00F7    277          BNEQ    30$                             ;IF NEQ NO
      00000123'EF   16  00F9    278          JSB     DECREF                          ;DECREMENT REFERENCE COUNT AND CALL DRIVER
                DA   11  00FF    279          BRB     30$                             ;
      50    0858 8F   3C  0101    280 40$:     MOVZWL  #SS$_DEVNOTALLOC,R0     ;SET DEVICE NOT ALLOCATED TO PROCESS
                18   11  0106    281 50$:     BRB     UNLOCK                          ;
```

```
                                0108    283          .SBTTL  LOCK I/O DATA BASE AND SEARCH FOR DEVICE
                                0108    284 ;
                                0108    285 ; SUBROUTINE TO LOCK I/O DATA BASE, SEARCH FOR DEVICE, AND CHECK FOR MAILBOX.
                                0108    286 ;
                                0108    287
                                0108    288 SEARCHDEV:                                  ;LOCK I/O DATA BASE AND SEARCH FOR DEVICE
-1                    51   04 AC   D0  0108    290          MOVL    DEVNAM(AP),R1      ;GET ADDRESS OF DEVICE NAME STRING DESCRIPTO
                           09   13  010C    291          BEQL    10$                ;IF EQL NONE
                      50   0C   3C  010E    292          MOVZWL  #SS$_ACCVIO,R0     ;SET ACCESS VIOLATION STATUS
                                0111    293          IFNORD  #8,(R1),UNLOCK     ;CAN DEVICE NAME DESCRIPTOR BE READ?
                           05   0117    294 10$:     RSB                        ;
                      50   24   3C  0118    295 NOPRIV:  MOVZWL  #SS$_NOPRIV,R0     ;SET NO PRIVILEGE
                           03   11  011B    296          BRB     UNLOCK             ;
                      50   01   3C  011D    297 NORMAL:  MOVZWL  #SS$_NORMAL,R0     ;SET NORMAL COMPLETION
                         FEDD'  31  C120    298 UNLOCK:  BRW     IOC$UNLOCK         ;UNLOCK I/O DATA BASE AND RETURN
```

```
                                0123   300          .SBTTL  DECREMENT REFERENCE COUNT, CLEAR OWNERSHIP, AND CALL DRIVER
                                0123   301 ;
                                0123   302 ; SUBROUTINE TO DECREMENT REFERENCE COUNT, CLEAR OWNERSHIP, AND CALL DRIVER
                                0123   303 ;
                                0123   304
-1                              0123   306 DECREF:                                                ;
                00 34 A1   17   E5  0123   307          BBCC    #DEV$V_ALL,UCB$L_DEVCHAR(R1),10$ ;CLEAR DEVICE ALLOCATED
                      50 A1    B7  0128   308 10$:     DECW    UCB$W_REFC(R1)                   ;DECREMENT REFERENCE COUNT
                      28 A1    D4  012B   309          CLRL    UCB$L_PID(R1)                    ;CLEAR OWNER PROCESS ID
                         2A    BB  012E   310          PUSHR   #^M<R1,R3,R5>                    ;SAVE REGISTERS
                   50    24 A1    D0  0130   311          MOVL    UCB$L_DDB(R1),R0                 ;GET ADDRESS OF DDB
                   50    0C A0    D0  0134   312          MOVL    DDB$L_DDT(R0),R0                 ;GET ADDRESS OF DDT
                      55    51    D0  0138   313          MOVL    R1,R5                            ;SET ADDRESS OF DEVICE UCB
                            52    D4  013B   314          CLRL    R2                               ;CLEAR CHANNEL NUMBER
                                  C13D   315          DSBINT  UCB$B_FIPL(R5)                   ;RAISE TO DRIVER FORK LEVEL
                   53    4C A5    D0  0144   316          MOVL    UCB$L_IRP(R5),R3                 ;GET ADDRESS OF CURRENT I/O PACKET
                   51    0C A0    D0  0148    .1          MOVL    DDT$L_CANCEL(R0), R1             ; Get Cancel entry point
                   06 51    10    E0  014C    .2          BBS     #16, R1, 20$                     ; +++++ Skip if Supervisor absolute
                      51    50    C0  0150    .3          ADDL2   R0, R1                           ; +++++ Convert to absolute
-2                    51    50    C2  0153   319          SUBL    R0,R1                            ;SUBTRACT OUT BASE ADDRESS
                            61    16  0156   320 20$:     JSB     (R1)                             ;CALL CANCEL I/O ROUTINE
                                  0158   321          ENBINT                                   ;ENABLE INTERRUPTS
                               2A BA  015B   322          POPR    #^M<R1,R3,R5>                    ;RESTORE REGISTERS
                                  05  015D   323          RSB                                      ;
                                      015E   324
```

```
                                015E    .2 .sbttl  DSX$MOUNT - Mount a device
                                015E    .3
                                015E    .4 ;++                                                                  [03]
                                015E    .5 ; Functional Description:                                            [03]
                                015E    .6 ;                                                                    [03]
                                015E    .7 ;       Calls the VMS $MOUNT service.                                [03]
                                015E    .8 ;                                                                    [03]
                                015E    .9 ; Inputs:                                                            [03]
                                015E   .10 ;                                                                    [03]
                                015E   .11 ;       4(AP) - Address of list of item identifiers                  [03]
                                015E   .12 ;                                                                    [03]
                                015E   .13 ; Implicit Inputs:                                                   [03]
                                015E   .14 ;                                                                    [03]
                                015E   .15 ;       DSA$GL_FLAGS                                                 [03]
                                015E   .16 ;                                                                    [03]
                                015E   .17 ; Outputs:                                                          [03]
                                015E   .18 ;                                                                    [03]
                                015E   .19 ; Implicit Outputs:                                                 [03]
                                015E   .20 ;                                                                    [03]
                                015E   .21 ; Side Effects:                                                     [03]
                                015E   .22 ;                                                                    [03]
                                015E   .23 ; Return Status:                                                    [03]
                                015E   .24 ;                                                                    [03]
                                015E   .25 ;       The return status from VMS is passed to the caller.          [03]
                                015E   .26 ;--                                                                  [03]
                                015E   .27
                         0000   015E   .28 .ENTRY  DSX$MOUNT, ^M<>          ;                                [03]
                           1C   E0  0160   .29          BBS     #DSA$V_USER, -  ; IF executing in standalone mode [03]
                0000FE00'EF         0162
                           05       0167   .30                                  DSA$GL_FLAGS,10$; THEN          [03]
                        50 01   D0  0168   .31          MOVL    #SS$_NORMAL, R0 ;       Set SS$_NORMAL return code. [03]
                           0A   11  016B   .32          BRB     20$             ;       Just return.            [03]
                                016D   .33 10$:                                 ; ELSE (user mode)              [03]
                           04 AC    DD  016D   .34          PUSHL   4(AP)           ;       Pass argument.          [03]
                           01       FB  0170   .35          CALLS   #1,G^SYS$MOUNT  ;       Call service.           [03]
                00000000'GF         0172
                                04  0177   .36 20$:    RET                     ; Return.                       [03]
                                    0178   .37
```

```
                                0178   .39 .sbttl  DSX$DISMOUNT - Dismount a device
                                0178   .40
                                0178   .41 ;++                                                                    [03]
                                0178   .42 ; Functional Description:                                             [03]
                                0178   .43 ;                                                                      [03]
                                0178   .44 ;       Calls the VMS $DISMOU service.                                 [03]
                                0178   .45 ;                                                                      [03]
                                0178   .46 ; Inputs:                                                            [03]
                                0178   .47 ;                                                                      [03]
                                0178   .48 ;       4(AP) - Addr. of char. string descriptor pointing to the       [03]
                                0178   .49 ;               device name                                            [03]
                                0178   .50 ;       8(AP) - Options flag                                           [03]
                                0178   .51 ;                                                                      [03]
                                0178   .52 ; Implicit Inputs:                                                    [03]
                                0178   .53 ;                                                                      [03]
                                0178   .54 ;       DSA$GL_FLAGS                                                   [03]
                                0178   .55 ;                                                                      [03]
                                0178   .56 ; Outputs:                                                           [03]
                                0178   .57 ;                                                                      [03]
                                0178   .58 ; Implicit Outputs:                                                   [03]
                                0178   .59 ;                                                                      [03]
                                0178   .60 ; Side Effects:                                                      [03]
                                0178   .61 ;                                                                      [03]
                                0178   .62 ; Return Status:                                                      [03]
                                0178   .63 ;                                                                      [03]
                                0178   .64 ;       The return status from VMS is passed to the caller.            [03]
                                0178   .65 ;--                                                                    [03]
                                0178   .66
                          0000  0178   .67 .ENTRY DSX$DISMOUNT, ^M<>          ;                                [03]
                    1C    E0    017A   .68          BBS     #DSA$V_USER, -    ; IF executing in standalone mode  [03]
0000FE00'EF                     017C
                    05          0181   .69                  DSA$GL_FLAGS,10$; THEN                               [03]
                50  01    D0    0182   .70          MOVL    #SS$_NORMAL, R0   ;       Set SS$_NORMAL return code.[03]
                    0D    11    0185   .71          BRB     20$               ;       Just return.               [03]
                                0187   .72 10$:                               ; ELSE (user mode)                 [03]
                  08 AC   DD    0187   .73          PUSHL   8(AP)             ;       Pass arguments.            [03]
                  04 AC   DD    018A   .74          PUSHL   4(AP)             ;                                  [03]
                    02    FB    018D   .75          CALLS   #2,G^SYS$DISMOU   ;       Call service.              [03]
00000000'GF                     018F
                          04    0194   .76 20$:     RET                       ; Return.                          [03]
                                0195   .77
                                0195   325         .END
```

```
$$ARGS                          =  00000002     D
$$T1                            =  0000000C     D
$ER                             =  00000001     D
$MODULE                            00000000 R   D   02
ALACMODE                        =  00000010     D
ALLOC$_ACMODE                   =  00000010     D
ALLOC$_DEVNAM                   =  00000004     D
ALLOC$_NARGS                    =  00000004     D
ALLOC$_PHYBUF                   =  0000000C     D
ALLOC$_PHYLEN                   =  00000008     D
DALLOC$_ACMODE                  =  00000008     D
DALLOC$_DEVNAM                  =  00000004     D
DALLOC$_NARGS                   =  00000002     D
DDB$B_ACPCLASS                     00000013     D
DDB$B_TYPE                         0000000A     D
DDB$C_LENGTH                       00000034     D
DDB$K_LENGTH                       00000034     D
DDB$L_ACPD                         00000010     D
DDB$L_DDT                          0000000C     D
DDB$L_LINK                         00000000     D
DDB$L_UCB                          00000004     D
DDB$T_DRVNAME                      00000024     D
DDB$T_NAME                         00000014     D
DDB$W_SIZE                         00000008     D
DDT$L_ALTSTART                     0000001C     D
DDT$L_CANCEL                       0000000C     D
DDT$L_FDT                          00000008     D
DDT$L_REGDUMP                      00000010     D
DDT$L_START                        00000000     D
DDT$L_UNITINIT                     00000018     D
DDT$L_UNSOLINT                     00000004     D
DDT$W_DIAGBUF                      00000014     D
DDT$W_ERRORBUF                     00000016     D
DECREF                             00000123 R   D   02
DEV$V_ALL                       =  00000017     D
DEV$V_SPL                       =  00000006     D
DEVNAM                          =  00000004     D
DLACMODE                        =  00000008     D
DS$AX_SOFTPCB                      ********     X   02
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSA$V_USER                      =  0000001C     D
DSX$DISMOUNT                       00000178 RG  D   02
DSX$MOUNT                          0000015E RG  D   02
EXE$ALLOC                          0000000A RG  D   02
EXE$DALLOC                         000000A3 RG  D   02
EXE$MAXACMODE                      ********     X   02
IOC$CVT_DEVNAM                     ********     X   02
IOC$GL_DEVLIST                     ********     X   02
IOC$SEARCHDEV                      ********     X   02
IOC$SEARCHGEN                      ********     X   02
IOC$UNLOCK                         ********     X   02
NOPRIV                             00000118 R   D   02
NORMAL                             0000011D R   D   02
PCB$B_ASTACT                       0000000C     D
PCB$B_ASTEN                        0000000D     D
PCB$B_PRI                          0000000B     D
PCB$B_PRIB                         0000002F     D
PCB$B_TYPE                         0000000A     D
PCB$B_WEFC                         0000002E     D
PCB$C_LENGTH                       0000008C     D
PCB$K_LENGTH                       0000008C     D
PCB$L_ARB                          00000084     D
PCB$L_ASTQBL                       00000014     D
PCB$L_ASTQFL                       00000010     D
PCB$L_EFC2P                        00000058     D
PCB$L_EFC3P                        0000005C     D
PCB$L_EFCS                         00000050     D
PCB$L_EFCU                         00000054     D
PCB$L_EFWM                         0000004C     D
PCB$L_JIB                          00000078     D
PCB$L_OWNER                        0000001C     D
PCB$L_PHD                          00000064     D
PCB$L_PHYPCB                       00000018     D
PCB$L_PID                          00000060     D
PCB$L_PQB                          0000004C     D
PCB$L_SQBL                         00000004     D
PCB$L_SQFL                         00000000     D
PCB$L_STS                          00000024     D
PCB$L_UIC                          00000088     D
PCB$L_WSSWP                        00000020     D
PCB$L_WTIME                        00000028     D
PCB$Q_PRIV                         0000007C     D
PCB$T_LNAME                        00000068     D
PCB$T_TERMINAL                     00000044     D
PCB$W_APTCNT                       00000030     D
PCB$W_ASTCNT                       00000038     D
PCB$W_BIOCNT                       0000003A     D
PCB$W_BIOLM                        0000003C     D
PCB$W_DIOCNT                       0000003E     D
PCB$W_DIOLM                        00000040     D
PCB$W_GPGCNT                       00000034     D
PCB$W_GRP                          0000008A     D
PCB$W_MEM                          00000088     D
PCB$W_MTXCNT                       0000000E     D
PCB$W_PPGCNT                       00000036     D
PCB$W_PRCCNT                       00000042     D
PCB$W_SIZE                         00000008     D
PCB$W_STATE                        0000002C     D
```

```
PCB$W_TMBU                00000032     D
PHYBUF                =   0000000C     D
PHYLEN                =   00000008     D
PR$_IPL               =   00000012     D
SEARCHDEV                 00000108 R   D   02
SIZ...                =   00000001     D
SS$_ACCVIO            =   0000000C     D
SS$_DEVALLOC          =   00000840     D
SS$_DEVALRALLOC       =   00000641     D
SS$_DEVNOTALLOC       =   00000858     D
SS$_IVDEVNAM          =   00000144     D
SS$_NOPRIV            =   00000024     D
SS$_NORMAL            =   00000001     D
SYS$DISMOU                ********     X   02
SYS$MOUNT                 ********     X   02
UCB$B_AMOD                00000053     D
UCB$B_CEX                 00000077     D
UCB$B_CM1                 0000004A     D
UCB$B_CM2                 0000004B     D
UCB$B_DEVCLASS            00000038     D
UCB$B_DEVTYPE             00000039     D
UCB$B_DIPL                00000052     D
UCB$B_DX_SCTCNT           000000A6     D
UCB$B_ERTCNT              00000070     D
UCB$B_ERTMAX              00000071     D
UCB$B_FEX                 00000076     D
UCB$B_FIPL                0000000B     D
UCB$B_LOCSRV              0000003C     D
UCB$B_OFFNDX              00000094     D
UCB$B_OFFRTC              00000095     D
UCB$B_REMSRV              0000003D     D
UCB$B_SECTORS             0000003C     D
UCB$B_SLAVE               00000074     D
UCB$B_SPR                 00000075     D
UCB$B_STATE               00000052     D
UCB$B_TRACKS              0000003D     D
UCB$B_TT_CRFILL           0000009D     D
UCB$B_TT_DECRF            000000A1     D
UCB$B_TT_DELFF            000000A2     D
UCB$B_TT_DESPEE           000000A0     D
UCB$B_TT_DETYPE           000000A4     D
UCB$B_TT_LFFILL           0000009E     D
UCB$B_TT_SPEED            0000009C     D
UCB$B_TYPE                0000000A     D
UCB$B_VERTSZ              0000003F     D
UCB$C_LENGTH              00000074     D
UCB$C_MB_LENGTH           00000090     D
UCB$C_TT_LENGTH           000000BC     D
UCB$K_LENGTH              00000074     D
UCB$K_MB_LENGTH           00000090     D
UCB$K_TT_LENGTH           000000BC     D
UCB$L_AMB                 00000054     D
UCB$L_ASTQBL              00000010     D
UCB$L_ASTQFL              0000000C     D
UCB$L_CPID                0000005C     D
UCB$L_CRB                 00000020     D
UCB$L_DDB                 00000024     D
UCB$L_DEVCHAR             00000034     D
UCB$L_DEVDEPEND           0000003C     D
UCB$L_DPC                 00000080     D
UCB$L_DUETIM              0000005C     D
UCB$L_DX_BFPNT            0000009C     D
UCB$L_DX_BUF              00000098     D
UCB$L_DX_RXDB             000000A0     D
UCB$L_EMB                 00000078     D
UCB$L_FIRST               00000014     D
UCB$L_FPC                 0000000C     D
UCB$L_FQBL                00000004     D
UCB$L_FQFL                00000000     D
UCB$L_FR3                 00000010     D
UCB$L_FR4                 00000014     D
UCB$L_IOQBL               00000044     D
UCB$L_IOQFL               00000040     D
UCB$L_IRP                 0000004C     D
UCB$L_LINK                0000002C     D
UCB$L_LOGADR              00000064     D
UCB$L_MAXBLOCK            00000084     D
UCB$L_MB_MBX              0000007C     D
UCB$L_MB_PORT             0000008C     D
UCB$L_MB_RAST             00000078     D
UCB$L_MB_SHB              00000080     D
UCB$L_MB_WAST             00000074     D
UCB$L_MB_WIOQBL           00000088     D
UCB$L_MB_WIOQFL           00000084     D
UCB$L_MEDIA               0000008C     D
UCB$L_NT_DATSSB           00000074     D
UCB$L_NT_INTSSB           00000078     D
UCB$L_OPCNT               00000060     D
UCB$L_OWNUIC              0000001C     D
UCB$L_PID                 00000028     D
UCB$L_RQBL                0000C004     D
UCB$L_RQFL                00000000     D
UCB$L_SVAPTE              00000068     D
UCB$L_SVPN                00000064     D
UCB$L_TT_DECHAR           000000A8     D
UCB$L_TT_RDUE             0000008C     D
UCB$L_TT_RTIMOU           000000B8     D
UCB$L_VCB                 00000030     D
UCB$T_PARTNER             0000000C     D
UCB$W_BCNT                0000006E     D
UCB$W_BCR                 00000096     D
UCB$W_BOFF                0000006C     D
UCB$W_BUFQUO              00000018     D
UCB$W_BYTESTOGO           0000003E     D
UCB$W_CHARGE              0000004A     D
UCB$W_CYLINDERS           0000003E     D
UCB$W_DA                  0000008C     D
UCB$W_DC                  0000008E     D
UCB$W_DEVBUFSIZ           0000003A     D
UCB$W_DEVSTS              0000005A     D
UCB$W_DIRSEQ              00000088     D
UCB$W_DSTADDR             00000018     D
UCB$W_DX_BCR              000000A4     D
UCB$W_ECT                 00000090     D
```

```
UCB$W_EC2                       00000092    D
UCB$W_ERRCNT                    00000072    D
UCB$W_FUNC                      0000007E    D
UCB$W_MB_SEED                   00000000    D
UCB$W_MSGCNT                    00000016    D
UCB$W_MSGMAX                    00000014    D
UCB$W_NT_CHAN                   0000007C    D
UCB$W_OFFSET                    0000008A    D
UCB$W_REFC                      00000050    D
UCB$W_SIZE                      00000008    D
UCB$W_SRCADDR                   0000001A    D
UCB$W_STS                       00000058    D
UCB$W_TT_DESIZE                 000000A5    D
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000C01A    D
UNLOCK                          00000120 R  D  02
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000195 (405.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000002	40.2 (1)	40.1 (1) 40.2 (1)
$$T1	=0000000C	40.2 (1)	40.1 (1) 40.2 (1)
$ER	=00000001	128.2 (1)	
$MODULE	00000000-R	128.2 (1)	
ALACMODE	=00000010	60 (1)	196 (1)
ALLOC$_ACMODE	=00000010	40.1 (1)	
ALLOC$_DEVNAM	=00000004	40.1 (1)	
ALLOC$_NARGS	=00000C04	40.1 (1)	
ALLOC$_PHYBUF	=0000000C	40.1 (1)	
ALLOC$_PHYLEN	=00000008	40.1 (1)	
DALLOC$_ACMODE	=00000008	40.2 (1)	
DALLOC$_DEVNAM	=00000004	40.2 (1)	
DALLOC$_NARGS	=00000002	40.2 (1)	
DDB$L_DDT	0000000C		#-312 (1)
DDB$L_LINK	00000000		265 (1) #-266 (1)
DDB$L_UCB	00000004		268 (1)
DDT$L_CANCEL	0000000C		#-316.1 (1)
DECREF	00000123-R	306 (1)	263 (1) 278 (1)
DEV$V_ALL	=00000017		#-187 (1) #-194 (1) #-275 (1) #-307 (1)
DEV$V_SPL	=00000006		#-166 (1)
DEVNAM	=00000004	66 (1)	#-290 (1)
DLACMODE	=00000008	67 (1)	
DS$AX_SOFTPCB	00000000-XR		128.4 (1) 247.1 (1)
DSA$GL_FLAGS	0000FE00		324.30 (1) 324.69 (1)
DSA$V_USER	=0000001C		#-324.29 (1) #-324.68 (1)
DSX$DISMOUNT	00000178-R	324.67 (1)	
DSX$MOUNT	0000015E-R	324.28 (1)	
EXE$ALLOC	0000000A-R	128.3 (1)	
EXE$DALLOC	000000A3-R	247 (1)	
EXE$MAXACMODE	00000000-XR		#-197 (1)
IOC$CVT_DEVNAM	00000000-XR		#-151 (1)
IOC$GL_DEVLIST	00000000-XR		265 (1)
IOC$SEARCHDEV	00000000-XR		#-253 (1)
IOC$SEARCHGEN	00000000-XR		#-134 (1)
IOC$UNLOCK	00000000-XR		#-200 (1) #-298 (1)
NOPRIV	00000118-R	295 (1)	#-170.1 (1)
NORMAL	0000011D-R	297 (1)	#-264 (1) #-267 (1)
PCB$L_PID	00000060		#-161 (1) #-193 (1) #-255 (1) #-271 (1)
PHYBUF	=0000000C	59 (1)	#-141 (1)
PHYLEN	=00000008	58 (1)	#-146.1 (1)
PR$_IPL	=00000012		#-315 (1) #-321 (1)
SEARCHDEV	00000108-R	288 (1)	#-132 (1) #-251 (1)
SS$_ACCVIO	=0000000C		#-182 (1) #-292 (1)
SS$_DEVALLOC	=00000840		#-163 (1)
SS$_DEVALRALLOC	=00000641		#-188 (1)
SS$_DEVNOTALLOC	=00000858		#-280 (1)
SS$_IVDEVNAM	=00000144		#-131 (1)
SS$_NOPRIV	=00000024		#-295 (1)
SS$_NORMAL	=00000001		#-185 (1) #-297 (1) #-324.31 (1) #-324.70 (1)
SYS$DISMOU	00000000-XR		324.75 (1)

```
SYS$MOUNT                       00000000-XR                  324.35 (1)
UCB$B_AMOD                      00000053                  #-198    (1)    #-273    (1)
UCB$B_FIPL                      0000000B                  #-315    (1)
UCB$L_DDB                       00000024                  #-311    (1)
UCB$L_DEVCHAR                   00000034                    166    (1)      187    (1)      194    (1)      275    (1)
                                                            307    (1)
UCB$L_IRP                       0000004C                  #-316    (1)
UCB$L_LINK                      0000002C                    268    (1)    #-269    (1)
UCB$L_PID                       00000028                  #-159    (1)    #-193    (1)    #-255    (1)    #-271    (1)
                                                          #-309    (1)
UCB$W_REFC                      00000050                  #-169    (1)    #-195    (1)    #-261    (1)    #-276    (1)
                                                          #-308    (1)
UNLOCK                          00000120-R    298    (1)  #-281    (1)    #-293    (1)    #-296    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ALLOCDEF	1	40.1 (1)	40.1 (1)
$DALLOCDEF	1	40.2 (1)	40.2 (1)
$DDBDEF	1	41 (1)	41 (1)
$DDTDEF	1	42 (1)	42 (1)
$DEFINI	1	41 (1)	41 (1) 42 (1) 43 (1) 44 (1) 45 (1) 46 (1) 47 (1) 48 (1) 49 (1) 49.2 (1)
$DEVDEF	1	43 (1)	43 (1)
$DS_DSADEF	5	49.2 (1)	49.2 (1)
$OFFDEF	1	40.1 (1)	40.1 (1) 40.2 (1)
$PCBDEF	4	44 (1)	44 (1)
$PRDEF	4	45 (1)	45 (1)
$PRVDEF	4	46 (1)	46 (1)
$PSLDEF	2	47 (1)	47 (1)
$SSDEF	21	48 (1)	48 (1)
$UCBDEF	10	49 (1)	49 (1)
DSBINT	1	315 (1)	315 (1)
ENBINT	1	321 (1)	321 (1)
IFNORD	1	293 (1)	293 (1)
MODNAM	1	128.2 (1)	128.2 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.08	00:00:00.25
Command processing	136	00:00:00.74	00:00:01.45
Pass 1	503	00:00:15.03	00:00:18.73
Symbol table sort	0	00:00:01.59	00:00:02.33
Pass 2	119	00:00:03.08	00:00:05.12
Symbol table output	33	00:00:00.23	00:00:00.56
Psect synopsis output	7	00:00:00.02	00:00:00.03
Cross-reference output	21	00:00:00.31	00:00:00.52
Assembler run totals	857	00:00:21.09	00:00:29.00

The working set limit was 1000 pages.
68759 bytes (135 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1024 non-local and 24 local symbols.
396 source lines were read in Pass 1, producing 0 object records in Pass 2.
71 pages of virtual memory were used to define 27 macros.

+---------------------------+
! Macro library statistics !
+---------------------------+

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	7
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	22

1301 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DEVALC/UPDA=(DEVALC.UPD,DEVALC.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(3) 62 DS$GPHARD Retreive address of P-table
(4) 117 DSP$GEN_PTABLES Setup UUT's to test

```
0000      1          .TITLE  DEVICE  *** DEVICE Handle PTABLEs
0000      2          .IDENT  /06-01/
0000      3          .LIST   MEB
0000      4          .NLIST  CND
0000      5          .DSABL  GBL
0000      6
0000      7 ;
0000      8 ; COPYRIGHT (C) 1977, 1980
0000      9 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     10 ;
0000     11 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     12 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     13 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     14 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     15 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     16 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     17 ; REMAIN IN DEC.
0000     18 ;
0000     19 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     20 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     21 ; CORPORATION.
0000     22 ;
0000     23 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     24 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     25
0000     26 ;++
0000     27 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     28 ;
0000     29 ; ABSTRACT:
0000     30 ;
0000     31 ; ENVIRONMENT:
0000     32 ;
0000     33 ; AUTHOR:       R. RIGGS        15-JAN-79       VERSION 01.
0000     34 ;               Dave Butenhof, 17-feb-1981, version 6.3
0000     35 ;       01      Fix linker truncation errors; change DSA$x_? references
0000     36 ;               to longword relative.
0000     37 ;--
```

```
                                0000     39 ;
                                0000     40 ; MACRO LIBRARIES:
                                0000     41 ;
                                0000     42
                                0000     43
                    00000001    0000     44 SS$_NORMAL=1
                                0000     48          $DS_HPODEF
                                0000     49          $DS_HDRDEF
                                0000     50          $DS_DSADEF
                                0000     51          $DS_DSDEF
                                0000     55
                                0000     56          .EXTRN  DS$GA_SELECTS, DS$GA_SELECTED, DS$GA_PTABLE
                                0000     57          .EXTRN  DS$PRINTF,      DS_ERRSUP,      DS$GPHARD
                                0000     58
                              0000C000   59          .PSECT  DATA, SHR,NOEXE, NOWRT, BYTE
                                0000     60          MODNAM  <DEVICE>
          45 43 49 56 45 44 00' 0000        $MODULE:        .ASCIC  \DEVICE\
                              06  0000
```

```
                                0007    62          .SBTTL  DS$GPHARD       Retreive address of P-table
                           00000000     63          .PSECT  CODE, SHR, NOWRT, EXE, BYTE
                                0000    64 ;++
                                0000    65 ; FUNCTIONAL DESCRIPTION:
                                0000    66 ;
                                0000    67 ;       Given a logical unit number, This routine returns the base
                                0000    68 ;       address of the associated P-table.
                                0000    69 ;
                                0000    70 ; CALLING SEQUENCE:
                                0000    71 ;
                                0000    72 ;       DS$GPHARD(UNIT,%REF(ADDRESS))
                                0000    73 ;
                                0000    74 ; INPUT PARAMETERS:
                                0000    75 ;
                                0000    76 ;       4(AP)   Logical unit number
                                0000    77 ;
                                0000    78 ; IMPLICIT INPUTS:
                                0000    79 ;
                                0000    80 ;       DS$GA_SELECTED  List of P-tables of units to be tested.
                                0000    81 ;
                                0000    82 ; OUTPUT PARAMETERS:
                                0000    83 ;
                                0000    84 ;       8(AP)   Address of longword to receive the address for this unit
                                0000    85 ;
                                0000    86 ; IMPLICIT OUTPUTS:
                                0000    87 ;
                                0000    88 ;       R1      Same as 8(AP)
                                0000    89 ;
                                0000    90 ; COMPLETION CODES:
                                0000    91 ;
                                0000    92 ;       SS$_NORMAL      If unit number was in range
                                0000    93 ;       DS$_ERROR       If unit number was too large (unsigned)
                                0000    94 ;
                                0000    95 ; SIDE EFFECTS:
                                0000    96 ;
                                0000    97 ;       R1       Returns with address of specified P-table
                                0000    98 ;--
                                0000    99
                           0004 0000   100 .ENTRY  DSX$GPHARD,^M<R2>
                                0002   101
51   00000000'EF    DE          0002   102         MOVAL   DS$GA_SELECTED,R1       ; Address table
 50    04 AC       01  C1      0009   103         ADDL3   #1,4(AP),R0             ; Change to origin 1
          61       50  D1      000E   104         CMPL    R0,(R1)                 ; Compare with length of select table
                   17  1A      0011   105         BGTRU   10$                     ; Branch if unit too large for table
0000FE0C'EF        50  D1      0013   106         CMPL    R0,L^DSA$GL_UNITS       ; Compare with number of units tested
                   0E  1A      001A   107         BGTRU   10$                     ; Branch if unit too large
        51    6140   D0      001C   108         MOVL    (R1)[R0],R1             ; Get address of P-table
      08 BC       51  D0      0020   109         MOVL    R1,@8(AP)               ; Store where user wants it
                   04  13      0024   110         BEQL    10$                     ; Branch if bogus
          50   01    D0      0026   111         MOVL    S^#SS$_NORMAL,R0        ; Set success
                   04  0029   112         RET                                     ; Return to user
                       002A   113 10$:
50   00660002 8F    D0      002A   114         MOVL    #DS$_ERROR,R0           ; Spaz
                   04  0031   115         RET
```

```
                     0032   117          .SBTTL  DSP$GEN_PTABLES Setup UUT's to test
                 00000032   118          .PSECT  CODE, SHR, NOWRT, EXE, BYTE
                     0032   119 ;++
                     0032   120 ; FUNCTIONAL DESCRIPTION:
                     0032   121 ;
                     0032   122 ;       This routine compares the unit types selected by the select
                     0032   123 ;       command with the device types the current program tests
                     0032   124 ;       and enters them in DS$GA_SELECTED.
                     0032   125 ;
                     0032   126 ; CALLING SEQUENCE:
                     0032   127 ;
                     0032   128 ;       CALLG   #0,DSP$GEN_PTABLES
                     0032   129 ;
                     0032   130 ; INPUT PARAMETERS:     NONE
                     0032   131 ;
                     0032   132 ; IMPLICIT INPUTS:
                     0032   133 ;
                     0032   134 ;       DS$GA_PTABLES, DS$GA_SELECTS
                     0032   135 ;       PROGRAM HEADER
                     0032   136 ;
                     0032   137 ; OUTPUT PARAMETERS:    NONE
                     0032   138 ;
                     0032   139 ; IMPLICIT OUTPUTS:
                     0032   140 ;
                     0032   141 ;       The P-TABLE's for the program
                     0032   142 ;
                     0032   143 ;--
```

```
                          007C  0032   145 .ENTRY  DSP$GEN_PTABLES,^M<R2,R3,R4,R5,R6>
                                0034   146
            0000FF0C'EF    D4   0034   147         CLRL    L^DSA$GL_UNITS          ; Clear count of units to test
            00000220'EF    D5   003A   148         TSTL    L$L_UNIT
                     65    13   0040   149         BEQL    60$                     ; EXIT IF NOT TESTING ANY UNITS
        56  00000000'EF    DE   0042   150         MOVAL   DS$GA_PTABLE,R6         ; Point to P-table list
                 55  66    D0   0049   151         MOVL    (R6),R5                 ; Get count of entries
    50 00000000'EF   55    E1   004C   152 10$:    BBC     R5,DS$GA_SELECTS,50$    ; Branch if not selected
                           0054   153
        54  0000021C 9F     D0   0054   154         MOVL    @#L$A_DEVP,R4           ; Get address of DEVTYP in program
                     4A    13   005B   155         BEQL    60$                     ; Branch if none
                  53  84    D0   005D   156         MOVL    (R4)+,R3                ; Get count of entries
                     45    13   0060   157         BEQL    60$                     ; Branch if none
                           0062   158 20$:
                  51  84    D0   0062   159         MOVL    (R4)+,R1                ; Get address of ASCIC devtype
                  50  81    9A   0065   160         MOVZBL  (R1)+,R0                ; Get length of devtype
               52   6645    D0   0068   161         MOVL    (R6)[R5],R2             ; Get address of P-table
                     36    13   006C   162         BEQL    50$                     ; Branch if no P-table
               52   26 A2   9E   006E   163         MOVAB   HP$T_TYPE(R2),R2        ; Point to type
                  50  82    91   0072   164         CMPB    (R2)+,R0                ; Compare string lengths
                     2A    12   0075   165         BNEQ    40$                     ; Branch if no match
                  82  81    91   0077   166 30$:    CMPB    (R1)+,(R2)+             ; Compare
                     25    12   007A   167         BNEQ    40$                     ; Branch if not the same
                    F8 50   F5   007C   168         SOBGTR  R0,30$                  ; Branch if more to compare
    50  0000FE0C'EF   01    C1   007F   169         ADDL3   #1,L^DSA$GL_UNITS,R0    ; Get next unit number
        00000220'EF   50    D1   0087   170         CMPL    R0,L$L_UNIT             ; Program test enough units?
                     17    14   008E   171         BGTR    60$                     ; Already selected enough units
            0000FE0C'EF    D6   0090   172         INCL    L^DSA$GL_UNITS          ; Update units being tested
 00000000'EF40   6645      D0   0096   173         MOVL    (R6)[R5],DS$GA_SELECTED[R0] ; Insert P-table as new unit
                     03    11   009F   174         BRB     50$                     ; Branch since match found
                           00A1   175 40$:
                  BE 53    F5   00A1   176         SOBGTR  R3,20$                  ; Count and continue in DEVTYP list
                           00A4   177 50$:
                  A5 55    F5   00A4   178         SOBGTR  R5,10$                  ; Next P-table
                           00A7   179 60$:
                     04    00A7   180         RET                             ; Return
                           00A8   181
                           00A8   182         .END
```

```
$ER                             = 00000001     D
$MODULE                           00000000 R   D   02
BIT...                          = 00660130     D
DS$GA_PTABLE                      ********     X   00
DS$GA_SELECTED                    ********     X   00
DS$GA_SELECTS                     ********     X   00
DS$GPHARD                         ********     X   00
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$PRINTF                         ********     X   00
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INSFMEM                     = 00660050     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSP$GER_PTABLES                   00000032 RG  D   03
DSX$GPHARD                        00000000 RG  D   03
DS_ERRSUP                         ********     X   00
HP$A_DEPENDENT                    00000032     D
HP$A_DEVICE                       00000018     D
HP$A_DVA                          0000001C     D
HP$A_LINK                         00000020     D
HP$B_DRIVE                        0000C00B     D
HP$B_FLAGS                        0000000A     D
HP$Q_DEVICE                       00000000     D
HP$T_DEVICE                       0000000C     D
HP$T_TYPE                         00000026     D
HP$W_SIZE                         00000008     D
HP$W_VECTOR                       00000024     D
L$A_CCP                           00000240     D
L$A_DEVP                          0000021C     D
L$A_DREG                          00000224     D
L$A_DTP                           00000218     D
L$A_ICP                           0000023C     D
L$A_LASTADR                       00000214     D
L$A_NAME                          00000208     D
L$A_REPP                          00000244     D
L$A_SECNAM                        00000250     D
L$A_STATAB                        00000248     D
L$A_TSTCNT                        00000254     D
L$L_ENVIRON                       00000204     D
L$L_ERRTYP                        0000024C     D
L$L_HEADLENGTH                    00000200     D
L$L_REV                           0000020C     D
L$L_UNIT                          00000220     D
L$L_UNUSED                        00000228     D
L$L_UPDATE                        00000210     D
SIZ...                          = 00000001     D
SS$_NORMAL                      = 00000001     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000007 (7.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	000000A8 (168.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	60 (2)	
$MODULE	00000000-R	60 (2)	
BIT...	=00660130	51 (2)	51 (2)
DS$GA_PTABLE	00000000-XR		150 (5) 56 (2)
DS$GA_SELECTED	00000000-XR		102 (3) #-173 (5) 56 (2)
DS$GA_SELECTS	00000000-XR		152 (5) 56 (2)
DS$GPHARD	00000000-XR		57 (2)
DS$K_ERROR	=00000002	51 (2)	
DS$K_NORMAL	=00000001	51 (2)	
DS$K_SEVERE	=00000004	51 (2)	
DS$K_SUBSYS	=00000066	51 (2)	51 (2)
DS$K_WARNING	=00000000	51 (2)	
DS$PRINTF	00000000-XR		57 (2)
DS$_ARITH	=006600D0	51 (2)	
DS$_ASBE	=00660118	51 (2)	
DS$_BADLINK	=006600F0	51 (2)	
DS$_BADTYPE	=006600E8	51 (2)	
DS$_BIIC	=00660120	51 (2)	
DS$_CHME	=006600A8	51 (2)	
DS$_CHMK	=006600E0	51 (2)	
DS$_DEVNAME	=00660108	51 (2)	
DS$_ERROR	=00660002	51 (2)	#-114 (3)
DS$_FHWE	=00660068	51 (2)	
DS$_FRAGBUF	=00660080	51 (2)	
DS$_ICBUSY	=006600C8	51 (2)	
DS$_ICERR	=006600C0	51 (2)	
DS$_IHWE	=00660060	51 (2)	
DS$_ILLCHAR	=00660018	51 (2)	
DS$_ILLPAGCNT	=00660078	51 (2)	
DS$_ILLUNIT	=00660100	51 (2)	
DS$_INSFMEM	=00660050	51 (2)	
DS$_IPL2HI	=006600B8	51 (2)	
DS$_IVADDR	=00660040	51 (2)	
DS$_IVVECT	=00660038	51 (2)	
DS$_KRNLSTK	=00660090	51 (2)	
DS$_LOGIC	=00660070	51 (2)	
DS$_MCHK	=00660088	51 (2)	
DS$_MMOFF	=00660058	51 (2)	
DS$_NEEDUNIT	=006600F8	51 (2)	
DS$_NODE	=00660128	51 (2)	
DS$_NOPCS	=00660110	51 (2)	
DS$_NORMAL	=00660001	51 (2)	
DS$_NOSUPPORT	=006600B1	51 (2)	
DS$_NOTDON	=00660030	51 (2)	
DS$_NOTIMP	=006600B0	51 (2)	51 (2)
DS$_NULLSTR	=00660010	51 (2)	
DS$_OVERFLOW	=00660008	51 (2)	
DS$_POWER	=00660098	51 (2)	
DS$_PROGERR	=00360020	51 (2)	
DS$_SEVERE	=00660004	51 (2)	

```
DS$_TRANSL                      =006600A0      51      (2)
DS$_TRUNCATE                    =00660028      51      (2)
DS$_UNEXPINT                    =006600D8      51      (2)
DS$_VASFULL                     =00660048      51      (2)
DS$_WARNING                     =00660000      51      (2)       51      (2)
DSA$GL_UNITS                     0000FE0C                      #-106   (3)     #-147   (5)     #-169   (5)     #-172   (5)
DSP$GEN_PTABLES                  00000032-R     145     (5)
DSX$GPHARD                       00000000-R     100     (3)
DS_ERRSUP                        00000000-XR                   57      (2)
HP$T_TYPE                        00000026                      163     (5)
L$A_DEVP                         0000021C                    #-154     (5)
L$L_UNIT                         00000220                    #-148     (5)     #-170   (5)
SS$_NORMAL                      =00000001      44      (2)    #-111     (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...					
$DEF	1	51 (2)						
$DEFINI	1	48 (2)	48 (2)	49 (2)	50 (2)			
$DS_DSADEF	5	50 (2)	50 (2)					
$DS_DSDEF	2	51 (2)	51 (2)					
$DS_HDRDEF	2	49 (2)	49 (2)					
$DS_HPODEF	2	48 (2)	48 (2)					
$EQU	1	51 (2)	51 (2)					
$EQULS1	1	51 (2)	51 (2)					
$EQULST	1	51 (2)	51 (2)					
$GBLINI	?		51 (2)					
$VIELD1	1	51 (2)						
MODNAM	1	60 (2)	60 (2)					

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.10	00:00:00.28
Command processing	139	00:00:00.73	00:00:01.59
Pass 1	355	00:00:04.62	00:00:10.60
Symbol table sort	0	00:00:00.21	00:00:00.28
Pass 2	77	00:00:00.76	00:00:00.99
Symbol table output	3	00:00:00.08	00:00:00.08
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	29	00:00:00.26	00:00:00.26
Assembler run totals	646	00:00:06.81	00:00:14.13

The working set limit was 950 pages.
22860 bytes (45 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 165 non-local and 7 local symbols.
182 source lines were read in Pass 1, producing 0 object records in Pass 2.
43 pages of virtual memory were used to define 15 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYS0.SYSMAINT]DS.MLB;218	1
DMA1:[SYS0.SYSMAINT]DIAG.MLB;953	4
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	10

241 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DEVICE/UPDA=(DEVICE.UPD,DEVICE.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

```
0001  0     %title '*** DIRECTORY Handle DIRECTORY command'
0002  0     module directory (                          ! Supervisor DIRECTORY command
0003  0                     ident = '07-17',
0004  0                     Debug,
0005  0                     OptLevel = 3
0006  0                     ) =
0007  0     !
0008  0     ! Copyright (c) 1980, 1982, 1983, 1984
0009  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0010  0     !
0011  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0012  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0013  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0014  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0015  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0016  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0017  0     ! REMAIN IN DEC.
0018  0     !
0019  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0021  0     ! CORPORATION.
0022  0     !
0023  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0024  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0025  0     !
0026  0
0027  0     !++
0028  0     ! FACILITY:
0029  0     !       DIAGNOSTIC SUPERVISOR TEST PROGRAM
0030  0     !
0031  0     ! ABSTRACT:
0032  0     !
0033  0     !       Read and print directories for supported mass storage volume formats:
0034  0     !       ODS-1/2 disks, ANSI magtapes, and RT-11 console media
0035  0     !
0036  0     !
0037  0     ! ENVIRONMENT:  DIAGNOSTIC SUPERVISOR
0038  0     !
0039  0     ! AUTHOR: Dave Butenhof         1-oct-1980
0040  0     !
0041  0     ! MODIFIED BY:
0042  0     !                       Dave Butenhof, 18-mar-1981, version 6.3
0043  0     !               01      Allow ODS subdirectory support under user mode.
0044  0     !                      -Dave butenhof, 02-Jun-1981, version 6.3
0045  0     !               02      Change library specs for more flexibility
0046  0     !                       - Dave butenhof, 26-Jun-1981, version 6.4
0047  0     !               03      Fix to use BOO$QIO instead of DSP$XX_READ (the latter does
0048  0     !                       not do error retries, nor does it handle errors well).
0049  0     !               04      - Dave Butenhof, 31-aug-1981, version 6.5
0050  0     !                       Support DQ device type (IDC disks) for Nebula.  Disks are
0051  0     !                       R80 and Rl02.
0052  0     !               05      - dave butenhof, 16-Sep-1981, version 6.5
0053  0     !                       Fix directory so unattached device results in 'nice' error
0054  0     !                       message, not general RMS$_DEV.
0055  0     !
0056  0     !               06      - Jack Stansbury, 22-Oct-1981, Version 6.-
0057  0     !                       Fixed a truncation error by changing the PSECT declarations to
```

```
;    0058  0     !                   conform to the .MAR file's PSECT's.
;    0059  0     !
;    0060  0     !           07      -- Dave Butenhof, 26-Oct-1981, Version 6.-
;    0061  0     !                   Alter call to LOC$PTABLE to include adapter address
;    0062  0     !                   argument.
;    0063  0     !
;    0064  0     !           08      - Dave Butenhof, 12-Jan-1982, Version 6.6
;    0065  0     !                   Set up Control C trap in DIRECTORY routines; this will allow
;    0066  0     !                   closing of any open files.
;    0067  0     !
;    0068  0     !           09      - Dave Butenhof, 25-Jan-1982, Version 6.6
;    0069  0     !                   Use DSR$ALLOCATE_PSEUDO_RPB routine instead of explicit init.
;    0070  0     !
;    0071  0     !           10      - Dave Butenhof, 03-Feb-1982, Version 6.6
;    0072  0     !                   Add ability to check filespec against (wildcarded) argument
;    0073  0     !                   before typing.  This is almost free, since the checking
;    0074  0     !                   routine exists in HELP already for checking keywords.  All
;    0075  0     !                   we need here is minimal pre-parsing to add 'implicit
;    0076  0     !                   wildcards', and the code to call the compare routine.
;    0077  0     !       [11] Dave Butenhof, 15-Apr-1982
;    0078  0     !            Add UDA50 disk support.
;    0079  0     !       [12] Dave Butenhof, 24-May-1982
;    0080  0     !            Add TM78 support (by removing special case code which disallows TU78)
;    0081  0     !
;    0082  0     !           13      M. Baggett,     9-March-1983, Version 6.11
;    0083  0     !                   Added support for CI-HSC50 and RC25/RCF25.
;    0084  0     !
;    0085  0     !           14      Bob Bergazzi    May 17, 1983    Version 6.11
;    0086  0     !                   Changed the order of searching the libraries.
;    0087  0     !           15      M. Baggett      11-Oct-1983     Version 6.13
;    0088  0     !                   Added support for CI-HSC50 magtape.
;    0089  0     !
;    0090  0     !           16      Bob Bergazzi    April 13, 1984  Version 7.0
;    0091  0     !                   Added support for ZZZ console storage device in ODS format.
;    0092  0     !                   to DSX$DIRECTORY routine.
;    0093  0     !
;    0094  0     !           17      Bob Bergazzi    May 21, 1984    Version 7.0
;    0095  0     !                   Added support for DIR/WIDE, which displays the directory
;    0096  0     !                   1 filename per line. This avoids the truncation of
;    0097  0     !                   long filenames under VMS V4.
;    0098  0     !
;    0099  0     !           18      RE MUSE         22 MAY,1984     Version 7.0
;    0100  0     !                   increased buffer size to handle long filenames for VMS v4
;    0101  0     !--
;    0102  0
;    0103  1     begin
;    0104  1     !
;    0105  1     ! Table of contents:
;    0106  1     !
;    0107  1     !
;    0108  1     ! Include files:
;    0109  1     !
;    0110  1
;    0111  1     library '$diag';                !
;    0112  1
;    0113  1     library '$ds';                  !
;    0114  1
```

```
;    0115  1     library 'sys$library:lib';      !
;    0116  1
;    0117  1     !
;    0118  1     ! Macros:
;    0119  1     !
;    0120  1
;    0121  1
;    0122  1     !
;    0123  1     ! psect definitions:
;    0124  1     !
;    0125  1
;    0126  1     psect
;    0127  1         plit  = data ( noexecute, share, nowrite, addressing_mode (long_relative));                !
;    0128  1
;    0129  1     psect
;    0130  1         own   = work ( noexecute, noshare,  write, addressing_mode (long_relative));               !
;    0131  1
;    0132  1     psect
;    0133  1         global = work ( noexecute, noshare,  write, addressing_mode (long_relative));              !
;    0134  1
;    0135  1     psect
;    0136  1         code   = code ( execute, share, nowrite, addressing_mode(long_relative));                  !
;    0137  1
;    0138  1     !
;    0139  1     ! equated symbols:
;    0140  1     !
;    0141  1     $ds_dsadef;                                     ! Define DSA locations
;    0142  1
;    0143  1     MAP
;    0144  1         DSA$GL_SID : BLOCK;                                                                        !
;    0145  1
;    0146  1     bind
;    0147  1         f_line = $ascic ('!AD!/');
;    0148  1
;    0149  1     linkage
;    0150  1         jsb_fake = jsb : nopreserve (2),
;    0151  1         jsb_scan = jsb (register = 3, register = 4, register = 5) : nopreserve (2);
;    0152  1
;    0153  1     !
;    0154  1     ! Own storage:
;    0155  1     !
;    0156  1         Own                                         !                                              [08]
;    0157  1             FileSpecMatch : Block [8, Byte],        !                                              [10]
;    0158  1             ControlFlag : Byte;                     !                                              [08]
;    0159  1
;    0160  1         GLOBAL                                      !                                              [17]
;    0161  1             WIDE_FLAG : BYTE;                       ! A 1 value indicates /WIDE was typed, 0 value = no /WIDE      [17]
;    0162  1
;    0163  1     !
;    0164  1     ! Literals                                      !                                              [16]
;    0165  1     !
;    0166  1
;    0167  1     LITERAL
;    0168  1             PR$_SID_TYPZZZ = 5;                     !                                              [16]
;    0169  1
;    0170  1
;    0171  1     !
```

```
;    0172  1      ! External references:
;    0173  1      !
;    0174  1
;    0175  1      external literal
;    0176  1          def$c_bkstp_len;                                ! Length of file spec backup
;    0177  1
;    0178  1      external
;    0179  1          begin_bliss : addressing_mode (long_relative),      ! Address of BEGIN code
;    0180  1          ds$gl_clibase : addressing_mode (long_relative),
;    0181  1          boo$select,                                     ! Non-interrupt "QIO" package stuff
;    0182  1          boo$al_vector : addressing_mode (long_relative),
;    0183  1          def$q_backstop : addressing_mode (long_relative),   ! "last-chance" file spec backup string
;    0184  1          def$q_dev : bblock addressing_mode (long_relative), ! Descriptor of system default device
;    0185  1          def$q_dir : bblock addressing_mode (long_relative); ! Descriptor of system default directory
;    0186  1
;    0187  1      external routine
;    0188  1          boo$qio : addressing_mode (long_relative),
;    0189  1          scan$numeric : jsb_scan addressing_mode (long_relative),
;    0190  1          kb_check : jsb_none addressing_mode (long_relative),
;    0191  1          script$flush : jsb_none addressing_mode (long_relative),
;    0192  1          init_context : jsb_none addressing_mode (long_relative),
;    0193  1          ds_cleanup : jsb_none addressing_mode (long_relative),
;    0194  1          qio$cleanup,
;    0195  1          breakup_file,
;    0196  1          Dsr$Key_Equal,                          ! compare two keywords                          [10]
;    0197  1          loc$ptable : jsb_loc$ptable addressing_mode (long_relative),    !                      [05]
;    0198  1          Dsr$Allocate_Pseudo_RPB : Addressing_Mode (Long_Relative),  !                           [09]
;    0199  1          Dsr$DeAllocate_Pseudo_RPB : Addressing_Mode (Long_Relative),    !                       [09]
;    0200  1          dsr$completion : jsb_0;
;    0201  1
```

```
;    0202  1     %Sbttl  'Put filename to buffer, if matches pattern'
;    0203  1     ;
;    0204  1     Routine PutBuffer (Count, Buffer, Desc) : NoValue =
;    0205  2         Begin
;    0206  2
;    0207  2     !+
;    0208  2     ! Function:
;    0209  2     !       Includes a filename within the printout buffer, if it matches the
;    0210  2     !       specified pattern, or if there is no pattern.
;    0211  2     !
;    0212  2     !       Count           Address of count longword (number of files typed)
;    0213  2     !       Buffer          Address of directory line buffer descriptor
;    0214  2     !       Desc            Address of file name buffer descriptor (at end of
;    0215  2     !                       'Buffer' buffer.
;    0216  2     !
;    0217  2     !-
;    0218  2
;    0219  2         Local                                   !                                                       [10]
;    0220  2             Wild;                               ! Dummy for key comparison routine                      [10]
;    0221  2
;    0222  2         Map                                     !                                                       [10]
;    0223  2             Buffer : Ref Block [, Byte],        !                                                       [10]
;    0224  2             Desc : Ref Block [, Byte];          !                                                       [10]
;    0225  2
;    0226  2         If .FileSpecMatch [Dsc$W_Length] Eql 0      ! If no pattern                                     [10]
;    0227  2          Or Dsr$Key_Equal (.Desc, FileSpecMatch, Wild)      ! .. or pattern matches                     [10]
;    0228  2         Then                                    !                                                       [10]
;    0229  3             Begin                               !                                                       [10]
;    0230  3             .Count = ..Count + 1;               ! Count this file                                       [10]
;    0231  3
;    C232  3             IF .WIDE_FLAG                                                   ! DIR/WIDE                  [17]
;    0233  4             THEN BEGIN                                                      !                           [17]
;    0234  4                     Buffer [Dsc$W_Length] = .Buffer [Dsc$W_Length] + .Desc [Dsc$W_Length];      !       [17]
;    0235  4                     $Ds_PrintF ($Ascic ('!AS!/'), Buffer [Dsc$W_Length]);   ! Print out buffer          [17]
;    0236  4                     Buffer [Dsc$W_Length] = 0;                              ! Reset buffer length       [17]
;    0237  4                     Ch$Fill (%C' ', 132, .Buffer [Dsc$A_Pointer])           ! Fill with blanks          [17]
;    0238  4                 END
;    0239  3             ELSE If (Buffer [Dsc$W_Length] = .Buffer [Dsc$W_Length] + .Desc [Dsc$W_Length]) Gtr 60  !   [10]
;    0240  3                 Then                            !                                                       [10]
;    0241  4                     Begin                       !                                                       [10]
;    0242  4                     $Ds_PrintF ($Ascic ('!AS!/'), Buffer [Dsc$W_Length]);       ! Print out buffer  [10]
;    0243  4                     Buffer [Dsc$W_Length] = 0;  ! Reset buffer length
;    0244  4                     Ch$Fill (%C' ', 80, .Buffer [Dsc$A_Pointer]);       ! Fill with blanks          [10]
;    0245  4                     End                         !                                                       [10]
;    0246  4
;    0247  3             End                                 !                                                       [10]
;    0248  3
;    0249  1         End;                                    !                                                       [10]


                                                        .TITLE  DIRECTORY *** DIRECTORY Handle DIRECTORY comman
                                                                                                              d
;                                                       .IDENT  \07-17\
                                                        .PSECT  WORK,NOEXE,2

                                                00000 FILESPECMATCH:
```

```
                                                        .BLKB   8
                                                00008 CONTROLCFLAG:
                                                        .BLKB   1
                                                00009 WIDE_FLAG::
                                                        .BLKB   1

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                2F  21  44  41  21  05  00000 P.AAA:   .ASCII  <5>\!AD!/\                                ;
                                2F  21  53  41  21  05  00006 P.AAB:   .ASCII  <5>\!A.!/\                                ;
                                2F  21  53  41  21  05  0000C P.AAC:   .ASCII  <5>\!AS!/\                                ;

                                                DSA$AL_APTMAIL=     65024
                                                DSA$GL_FLAGS=       65024
                                                DSA$GL_APTLOM=      65028
                                                DSA$GL_PASSES=      65032
                                                DSA$GL_UNITS=       65036
                                                DSA$GL_SECTNO=      65040
                                                DSA$GL_SID=         65044
                                                DSA$GL_MSGTYP=      65088
                                                DSA$GL_ERRNO=       65092
                                                DSA$GL_EVENT=       65096
                                                DSA$GL_SUBTNO=      65100
                                                DSA$GL_TESTNO=      65104
                                                DSA$GL_PASSNO=      65108
                                                DSA$GL_DEVLEN=      65112
                                                DSA$GL_DEVNAM=      65116
                                                DSA$GL_MSGPTR=      65128
                                                F_LINE=             P.AAA
                                                        .EXTRN  DEF$C_BKSTP_LEN
                                                        .EXTRN  BEGIN_BLISS, DS$GL_CLIBASE
                                                        .EXTRN  BOO$SELECT, BOO$AL_VECTOR
                                                        .EXTRN  DEF$Q_BACKSTOP, DEF$Q_DEV
                                                        .EXTRN  DEF$Q_DIR, BOO$QIO
                                                        .EXTRN  SCAN$NUMERIC, KB_CHECK
                                                        .EXTRN  SCRIPT$FLUSH, INIT_CONTEXT
                                                        .EXTRN  DS_CLEANUP, QIO$CLEANUP
                                                        .EXTRN  BREAKUP_FILE, DSR$KEY_EQUAL
                                                        .EXTRN  LOC$PTABLE, DSR$ALLOCATE_PSEUDO_RPB
                                                        .EXTRN  DSR$DEALLOCATE_PSEUDO_RPB
                                                        .EXTRN  DSR$COMPLETION, DS$PRINTF

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                            01FC 00000 PUTBUFFER:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8                               ; 0204
                                58 00000000G    9F  9E 00002    MOVAB   @#DS$PRINTF, R8                                 ;
                                57 00000000'    EF  9E 00009    MOVAB   FILESPECMATCH, R7                               ;
                                5E              04  C2 00010    SUBL2   #4, SP                                          ;
                                                67  B5 00013    TSTW    FILESPECMATCH                                   ; 0226
                                                0F  13 00015    BEQL    1$                                              ;
                                      4080      8F  BB 00017    PUSHR   #^M<R7,SP>                                      ; 0227
                                        0C      AC  DD 0001B    PUSHL   DESC                                            ;
                            0000G   CF          03  FB 0001E    CALLS   #3, DSR$KEY_EQUAL                               ;
                                    4E          50  E9 00023    BLBC    R0, 3$                                          ;
                                          04    BC  D6 00026 1$:  INCL    @COUNT                                          ; 0230
                                    56    08    AC  D0 00029    MOVL    BUFFER, R6                                      ; 0234
```

```
                                 1B        09    A7  E9 0002D           BLBC    WIDE_FLAG, 2$                                   ; 0232
                                 66        0C    BC  A0 00031           ADDW2   @DESC, (R6)                                     ; 0234
                                                 56  DD 00035           PUSHL   R6                                              ; 0235
                                    00000000'    EF  9F 00037           PUSHAB  P.AAB                                           ;
                                 68              02  FB 0003D           CALLS   #2, DS$PRINTF                                   ;
                                                 66  B4 00040           CLRW    (R6)                                            ; 0236
    0084    8F          20       6E              00  2C 00042           MOVC5   #0, (SP), #32, #132, @4(R6)                     ; 0237
                                           04    B6    00049
                                                 04  0004B             RET                                                     ;
                                 50              66  3C 0004C 2$:       MOVZWL  (R6), R0                                        ; 0239
                                 51        0C    BC  3C 0004F           MOVZWL  @DESC, R1                                       ;
                                 50              51  C0 00053           ADDL2   R1, R0                                          ;
                                 66              50  B0 00056           MOVW    R0, (R6)                                        ;
                                 3C              50  D1 00059           CMPL    R0, #60                                         ;
                                                 16  15 0005C           BLEQ    3$
                                                 56  DD 0005E           PUSHL   R6                                              0242
                                    00000000'    EF  9F 00060           PUSHAB  P.AAC                                           ;
                                 68              02  FB 00066           CALLS   #2, DS$PRINTF                                   ;
                                                 66  B4 00069           CLRW    (R6)                                            ; 0243
    0050    8F          20       6E              00  2C 0006B           MOVC5   #0, (SP), #32, #80, @4(R6)                      ; 0244
                                           04    B6    00072
                                                 04  00074 3$:         RET                                                     ; 0249

; Routine Size:  117 bytes,    Routine Base:  CODE + 0000
```

```
;    0250  1     %sbttl 'RAD50'
;    0251  1     ;
;    0252  1     routine rad50 (data, address) =
;    0253  2         begin
;    0254  2
;    0255  2     !++
;    0256  2     ! Functional description:
;    0257  2     !       Routine to convert RAD50 to ASCII
;    0258  2     !       writes the converted characters into a byte vector at ADDRESS
;    0259  2     !       Routine value is number of characters that were non blank
;    0260  2     !
;    0261  2     ! Formal parameters:
;    0262  2     !
;    0263  2     !       data    16 bit quantity containing 3 rad50 characters
;    0264  2     !       address address of byte vector to receive the 3 characters in ascii
;    0265  2     !
;    0266  2     ! Completion codes:     none
;    0267  2     !
;    0268  2     !--
;    0269  2
;    0270  2         map
;    0271  2             address : ref vector [3, byte];         ! Map out where to store data
;    0272  2
;    0273  2         local
;    0274  2             ptr;                                    ! Point to first space in string
;    0275  2
;    0276  2         bind
;    0277  2             xlate = uplit byte(%ascii' ABCDEFGHIJKLMNOPQRSTUVWXYZ$.?0123456789')
;    0278  2             : vector [40, byte];
;    0279  2
;    0280  2         address [0] = .xlate [.data/1600];
;    0281  2         address [1] = .xlate [(.data/40) mod 40];
;    0282  2         address [2] = .xlate [.data mod 40];
;    0283  2
;    0284  2         if ch$fail (ptr = ch$find_ch (3, ch$ptr (.address), %c' ')) then 3 else ch$diff (.ptr, ch$ptr (.address))
;    0285  2
;    0286  1         end;


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 20  00012 P.AAD:  .ASCII  \ ABCDEFGHIJKLMNOPQRSTUVWXYZ$.?0123456789\  ;
3F 2E 24 5A 59 58 57 56 55 54 53 52 51 50 4F  00021                                                            ;
               39 38 37 36 35 34 33 32 31 30  00030                                                            ;

                                                XLATE=              P.AAD


                                                        .PSECT  CODE,NOWRT,  SHR,2

                                          000C 00000 RAD50:  .WORD   Save R2,R3                                 ; 0252
                                53 00000000'  EF  9E 00002         MOVAB   XLATE, R3                          ;
                                52        08  AC  D0 00009         MOVL    ADDRESS, R2                        ; 0280
                     50    04   AC 00000640  8F  C7 0000D         DIVL3   #1600, DATA, R0                    ;
                                62           6340  90 00016         MOVB    XLATE[R0], (R2)                    ;
                     50    04   AC           28  C7 000'A         DIVL3   #40, DATA, R0                      ; 0281
```

```
                7E              00              50          01  7A 0001F          EMUL    #1, R0, #0, -(SP)                       ;
                50              50              8E          28  7B 00024          EDIV    #40, (SP)+, R0, R0                      ;
                                        01      A2        6340  90 00029          MOVB    XLATE[R0], 1(R2)                        ;
                7E              00      04      AC          01  7A 0002E          EMUL    #1, DATA, #0, -(SP)                     ; 0282
                50              50              8E          28  7B 00034          EDIV    #40, (SP)+, R0, R0                      ;
                                        02      A2        6340  90 00039          MOVB    XLATE[R0], 2(R2)                        ;
                                62              03          20  3A 0003E          LOCC    #32, #3, (R2)                           ; 0284
                                                            02  12 00042          BNEQ    1$                                      ;
                                                            51  D4 00044          CLRL    R1                                      ;
                                                            51  D5 00046 1$:      TSTL    PTR                                     ;
                                                            05  12 00048          BNEQ    2$                                      ;
                                                51          03  D0 0004A          MOVL    #3, R1                                  ;
                                                            03  11 0004D          BRB     3$                                      ;
                                                51          52  C2 0004F 2$:      SUBL2   R2, R1                                  ;
                                                50          51  D0 00052 3$:      MOVL    R1, R0                                  ; 0286
                                                                04 00055          RET                                             ;

; Routine Size:  86 bytes,    Routine Base:  CODE + 0075
```

```
;    0287  1     %sbttl 'format_ods1'
;    0288  1     routine format_ods1 (length, dir, buffer, count) : novalue =
;    0289  1
;    0290  1     !++
;    0291  1     ! Functional description:
;    0292  1     !
;    0293  1     !       This routine will format a ODS-1 directory record
;    0294  1     !
;    0295  1     ! Formal parameters:
;    0296  1     !
;    0297  1     !       length  length of record
;    0298  1     !       dir     Address of record
;    0299  1     !       buffer  Address of descriptor of output buffer
;    0300  1     !
;    0301  1     !       count   address of longword to get count of files
;    0302  1     !--
;    0303  1
;    0304  2         begin
;    0305  2
;    0306  2         local
;    0307  2             desc : block [8, byte],                 ! Desc. of area for fao output
;    0308  2             name : vector [16, byte],               ! ASCIC file name
;    0309  2             type : vector [4, byte];                ! ascic file type
;    0310  2
;    0311  2         map
;    0312  2             dir : ref block [, byte],               ! Address of record
;    0313  2             buffer : ref block [8, byte];           ! Address of buffer descriptor
;    0314  2
;    0315  2         if .dir [0, 0, 16, 0] neq 0
;    0316  2         then
;    0317  3             begin
;    0318  3
;    0319  3             if (.buffer [dsc$w_length] mod 20) neq 0
;    0320  3             then
;    0321  3                 buffer [dsc$w_length] = .buffer [dsc$w_length] + 20 - (.buffer [dsc$w_length] mod 20);
;    0322  3
;    0323  3             name [0] = rad50 (.dir [6, 0, 16, 0], name [1]) + rad50 (.dir [8, 0, 16, 0], name [4]) + rad50 (.dir
;    0324  3                 [10, 0, 16, 0], name [7]);
;    0325  3             type [0] = rad50 (.dir [12, 0, 16, 0], type [1]);
;    0326  3             desc [dsc$w_length] = 19;
;    0327  3             desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0328  3             $fao ($ascid ('!19<!AC.!AC;!UW!>'), desc, desc, name, type, .dir [14, 0, 16, 0]);       !               [10]
;    0329  3             PutBuffer (.Count, .Buffer, Desc)
;    0330  2             end;
;    0331  2
;    0332  1         end;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

57  55  21  3B  43  41  21  2E  43  41  21  3C  39  31  21  0003A P.AAF:  .ASCII  \!19<!AC.!AC;!UW!>\                    ;
                                                    3E  21  00049                                                       ;
                                                            0004B        .BLKB   1                                      ;
                                                00000011    0004C P.AAE:  .LONG   17                                    ;
                                                00000000'   00050        .ADDRESS P.AAF                                 ;
```

```
                                                        .EXTRN  SYS$FAO

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                003C 00000 FORMAT_ODS1:
                                                        .WORD   Save R2,R3,R4,R5                ; 0288
                                    55      A5  AF  9E 00002            MOVAB   RAD50, R5
                                    5E          1C  C2 00006            SUBL2   #28, SP
                                    53      08  AC  D0 00009            MOVL    DIR, R3             ; 0315
                                                63  B5 0000D            TSTW    (R3)
                                                01  12 0000F            BNEQ    1$
                                                    04 00011            RET
                                    54      0C  AC  D0 00012 1$:        MOVL    BUFFER, R4          ; 0319
                                    50          64  3C 00016            MOVZWL  (R4), R0
                7E          00      50          01  7A 00019            EMUL    #1, R0, #0, -(SP)
                50          50      8E          14  7B 0001E            EDIV    #20, (SP)+, R0, R0
                                                50  D5 00023            TSTL    R0
                                                0B  13 00025            BEQL    2$
                                    51          64  3C 00027            MOVZWL  (R4), R1            ; 0321
                                50  51          50  C3 0002A            SUBL3   R0, R1, R0
                                64  50          14  A1 0002E            ADDW3   #20, R0, (R4)
                                            05  AE  9F 00032 2$:        PUSHAB  NAME+1              ; 0323
                                    7E      06  A3  3C 00035            MOVZWL  6(R3), -(SP)
                                    65          02  FB 00039            CALLS   #2, RAD50
                                    52          50  D0 0003C            MOVL    R0, R2
                                            08  AE  9F 0003F            PUSHAB  NAME+4
                                    7E      08  A3  3C 00042            MOVZWL  8(R3), -(SP)
                                    65          02  FB 00046            CALLS   #2, RAD50
                                    52          50  C0 00049            ADDL2   R0, R2
                                            0B  AE  9F 0004C            PUSHAB  NAME+7              ; 0324
                                    7E      0A  A3  3C 0004F            MOVZWL  10(R3), -(SP)
                                    65          02  FB 00053            CALLS   #2, RAD50
                        04  AE              52      50  81 00056        ADDB3   R0, R2, NAME
                                            01  AE  9F 0005B            PUSHAB  TYPE+1              ; 0325
                                    7E      0C  A3  3C 0005E            MOVZWL  12(R3), -(SP)
                                    65          02  FB 00062            CALLS   #2, RAD50
                                    6E          50  90 00065            MOVB    R0, TYPE
                                14  AE          13  B0 00068            MOVW    #19, DESC           ; 0326
                                    50          64  3C 0006C            MOVZWL  (R4), R0            ; 0327
                                18  AE      04 B440 9E 0006F            MOVAB   @4(R4)[R0], DESC+4
                                    7E      0E  A3  3C 00075            MOVZWL  14(R3), -(SP)       ; 0328
                                            04  AE  9F 00079            PUSHAB  TYPE
                                            0C  AE  9F 0007C            PUSHAB  NAME
                                            20  AE  9F 0007F            PUSHAB  DESC
                                            24  AE  9F 00082            PUSHAB  DESC
                                        00000000' EF  9F 00085          PUSHAB  P.AAE
                        00000000G   00          06  FB 0008B            CALLS   #6, SYS$FAO
                                            14  AE  9F 00092            PUSHAB  DESC                ; 0329
                                                54  DD 00095            PUSHL   R4
                                            10  AC  DD 00097            PUSHL   COUNT
                                8B  A5          03  FB 0009A            CALLS   #3, PUTBUFFER
                                                    04 0009E            RET                         ; 0332

; Routine Size:  159 bytes,    Routine Base:  CODE + 00CB
```

```
;    0333  1     %sbttl 'format_ods2'
;    0334  1     routine format_ods2 (length, dir, buffer, count) : novalue =
;    0335  1
;    0336  1     !++
;    0337  1     ! Functional description:
;    0338  1     !
;    0339  1     !       This routine will format an ODS-2 directory record
;    0340  1     !
;    0341  1     ! Formal parameters:
;    0342  1     !
;    0343  1     !       length  length of record
;    0344  1     !       dir     Address of record
;    0345  1     !       buffer  Address of descriptor of output buffer
;    0346  1     !
;    0347  1     !       count   address of longword to get file count
;    0348  1     !--
;    0349  1
;    0350  2         begin
;    0351  2
;    0352  2         local
;    0353  2             desc : block [8, byte];
;    0354  2
;    0355  2         map
;    0356  2             dir : ref block [, byte],                   ! Address of record
;    0357  2             buffer : ref block [8, byte];               ! Address of buffer descriptor
;    0358  2
;    0359  2         bind
;    0360  2             ver_offset = -2 + dir$c_length + ((.dir [-2 + dir$b_namecount] + 1) and not 1);
;    0361  2
;    0362  2         incr index from ver_offset to .length - 1 by 8 do
;    0363  3             begin
;    0364  3             IF .WIDE_FLAG
;    0365  4                THEN BEGIN
;    0366  4                     if (.buffer [dsc$w_length] mod 132) neq 0
;    0367  4                        then
;    0368  4                         buffer [dsc$w_length] = .buffer [dsc$w_length] + 132 - (.buffer [dsc$w_length] mod 132);
;    0369  4                     desc [dsc$w_length] = 132;
;    0370  4                     desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0371  5                     $fao ($ascid ('!AC;!UW'), desc, desc, dir [-2 + dir$b_namecount], .dir [.index, 0, 16, 0])
;    0372  4                     END
;    0373  4                ELSE BEGIN
;    0374  4                     if (.buffer [dsc$w_length] mod 20) neq 0
;    0375  4                        then
;    0376  4                         buffer [dsc$w_length] = .buffer [dsc$w_length] + 20 - (.buffer [dsc$w_length] mod 20);
;    0377  4
;    0378  4                     desc [dsc$w_length] = 19;
;    0379  4                     desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0380  5                     $fao ($ascid ('!19<!AC;!UW!>'), desc, desc, dir [-2 + dir$b_namecount], .dir [.index, 0, 16, 0])
;    0381  3                     END;
;    0382  3             PutBuffer (.Count, .Buffer, Desc)
;    0383  2             end;
;    0384  2
;    0385  1         end;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2
```

```
                                57  55  21  3B  43  41  21   00054 P.AAH:   .ASCII  \!AC;!UW\                          ;
                                                             0005B         .BLKB   1
                                                 00000007    0005C P.AAG:  .LONG   7                                   ;
                                                 00000000'   00060         .ADDRESS P.AAH                            ;
        3E  21  57  55  21  3B  43  41  21  3C  39  31  21   00064 P.AAJ:   .ASCII  \!19<!AC;!UW!>\                    ;
                                                             00071         .BLKB   3
                                                 0000000D    00074 P.AAI:  .LONG   13                                  ;
                                                 00000000'   00078         .ADDRESS P.AAJ                            ;


                                                                           .PSECT  CODE,NOWRT,  SHR,2

                                                     003C 00000 FORMAT_ODS2:
                                                                           .WORD   Save R2,R3,R4,R5                  ; 0334
                                    5E                08  C2 00002         SUBL2   #8, SP                            ;
                                    52          08    AC  7D 00005         MOVQ    DIR, R2                           ; 0360
                                    50          03    A2  9A 00009         MOVZBL  3(R2), R0                         ;
                                                      50  D6 0000D         INCL    R0                                ;
                                    50                01  8A 0000F         BICB2   #1, R0                            ;
                                    54          04    A0  9E 00012         MOVAB   4(R0), R4                         ; 0362
                        55          04    AC            01  C3 00016         SUBL3   #1, LENGTH, R5                    ;
                                                    00A0  31  0001B         BRW     6$                                ; 0382
                                    46 00000000'    EF  E9 0001E 1$:      BLBC    WIDE_FLAG, 3$                     ; 0364
                                    50              63  3C 00025         MOVZWL  (R3), R0                          ; 0366
        7E          00              50              01  7A 00028         EMUL    #1, R0, #0, -(SP)                 ;
        50          50              8E 00000084     8F  7B 0002D         EDIV    #132, (SP)+, R0, R0               ;
                                                      50  D5 00036         TSTL    R0                                ;
                                                      0D  13 00038         BEQL    2$                                ;
                                    51                63  3C 0003A         MOVZWL  (R3), R1                          ; 0368
                        50          51                50  C3 0003D         SUBL3   R0, R1, R0                        ;
                        63          50          0084  8F  A1 00041         ADDW3   #132, R0, (R3)                    ;
                                    6E            84  8F  9B 00047 2$:      MOVZBW  #132, DESC                        ; 0369
                                    50                63  3C 0004B         MOVZWL  (R3), R0                          ; 0370
                              04    AE            04  B340  9E 0004E         MOVAB   @4(R3)[R0], DESC+4                ;
                                                      6442  9F 00054         PUSHAB  (INDEX)[R2]                       ; 0371
                                    7E                9E  3C 00057         MOVZWL  @(SP)+, -(SP)                     ;
                                                03    A2  9F 0005A         PUSHAB  3(R2)                             ;
                                                08    AE  9F 0005D         PUSHAB  DESC                              ;
                                                0C    AE  9F 00060         PUSHAB  DESC                              ;
                                     00000000'        EF  9F 00063         PUSHAB  P.AAG                             ;
                                                      3D  11 00069         BRB     5$                                ;
                                    50                63  3C 0006B 3$:      MOVZWL  (R3), R0                          ; 0374
        7E          00              50                01  7A 0006E         EMUL    #1, R0, #0, -(SP)                 ;
        50          50              8E                14  7B 00073         EDIV    #20, (SP)+, R0, R0                ;
                                                      50  D5 00078         TSTL    R0                                ;
                                                      0B  13 0007A         BEQL    4$                                ;
                                    51                63  3C 0007C         MOVZWL  (R3), R1                          ; 0376
                        50          51                50  C3 0007F         SUBL3   R0, R1, R0                        ;
                        63          50                14  A1 00083         ADDW3   #20, R0, (R3)                     ;
                                    6E                13  B0 00087 4$:      MOVW    #19, DESC                         ; 0378
                                    50                63  3C 0008A         MOVZWL  (R3), R0                          ; 0379
                              04    AE            04  B340  9E 0008D         MOVAB   @4(R3)[R0], DESC+4                ;
                                                      6442  9F 00093         PUSHAB  (INDEX)[R2]                       ; 0380
                                    7E                9E  3C 00096         MOVZWL  @(SP)+, -(SP)                     ;
                                                03    A2  9F 00099         PUSHAB  3(R2)                             ;
```

```
                                        08    AE   9F  0009C          PUSHAB  DESC                                                ;
                                        0C    AE   9F  0009F          PUSHAB  DESC                                                ;
                                  00000000'   EF   9F  000A2          PUSHAB  P.AAI                                               ;
                          00000000G     00          05  FB  000A8  5$:  CALLS   #5, SYS$FAO                                      ;
                                              4008    8F  BB  000AF          PUSHR   #^M<R3,SP>                                  ; 0382
                                                10    AC  DD  000B3          PUSHL   COUNT                                       ;
                                FDDB    CF            03  FB  000B6          CALLS   #3, PUTBUFFER                               ;
                                        54            08  C0  000BB          ADDL2   #8, INDEX                                   ;
                                        55            54  D1  000BE  6$:      CMPL    INDEX, R5                                   ;
                                                      03  14  000C1          BGTR    7$                                          ;
                                                  FF58    31  000C3          BRW     1$                                          ;
                                                          04  000C6  7$:      RET                                                 ; 0385

; Routine Size:  199 bytes,    Routine Base:  CODE + 016A
```

```
;    0386  1     %sbttl 'RT-11 directory'
;    0387  1     routine rt11_direct (unit, q_device, count) =
;    0388  2         begin
;    0389  2
;    0390  2         bind
;    0391  2             l = .q_device : word;
;    0392  2
;    0393  2         local
;    0394  2             segment,
;    0395  2             extra_length,
;    0396  2             address_first,
;    0397  2             RPB : Ref Block [, byte],       ! Fake RPB                                          [09]
;    0398  2             segblk : bblock [1024], ! Buffer to read segment
;    0399  2             t_buffer : bblock [80], ! Buffer for directory line
;    0400  2             name : vector [7, byte],        ! Buffer for ASCIC file name
;    0401  2             type : vector [4, byte],        ! Buffer for ASCIC file type
;    0402  2             OutBuffer : BBlock [Dsc$C_S_Bln],       ! Descriptor for buffer                 [10]
;    0403  2             q_buffer : bblock [dsc$c_s_bln];        ! Descriptor for FAO
;    0404  2
;    0405  2         RPB = Dsr$Allocate_Pseudo_RPB (BTD$K_Console, 0, 0, 0, 1, 0);       !                   [09]
;    0406  2         segment = 1;
;    0407  2         OutBuffer [Dsc$W_Length] = 0;
;    0408  2         OutBuffer [Dsc$A_Pointer] = T_Buffer;
;    0409  2         $ds_printf ($ascic ('!/!/Directory _!AS!/!/'), .q_device);
;    0410  2         ch$fill (%c' ', 80, t_buffer);
;    0411  2
;    0412  2         while 1 do
;    0413  3             begin
;    0414  3
;    0415  3             local
;    0416  3                 status;
;    0417  3
;    0418  3             if .segment eql 0 then exitloop;
;    0419  3
;    0420  4             if not (status = boo$qio (segblk,       ! Read directory segment                    [03]
;    0421  4                 1024,                       !  two blocks long                              [03]
;    0422  4                 4 + .segment*2,             !  from disk logical block                      [03]
;    0423  4                 io$_readlblk,               !                                               [03]
;    0424  4                 0,                          !  physical address                             [03]
;    0425  4                 .rpb))                      !  address of rpb                               [03]
;    0426  3             then return .status;            !                                               [03]
;    0427  3
;    0428  3             segment = .segblk [rt$w_segment];       ! Link to next segment
;    0429  3             extra_length = .segblk [rt$w_extralen]; ! Variable part size
;    0430  3             address_first = segblk [rt$w_first] - rt$_fixed - .extra_length;
;    0431  3
;    0432  3             while 1 do
;    0433  4                 begin
;    0434  4
;    0435  4                 bind
;    0436  4                     entry = (address_first = .address_first + rt$_fixed + .extra_length) : bblock;
;    0437  4
;    0438  4                 if .entry [rt$v_endseg] then exitloop;      ! End of segment
;    0439  4
;    0440  4                 if .entry [rt$v_perm]               ! Only list permanent files
;    0441  4                 then
;    0442  5                     begin
```

```
;    0443  6                    name [0] = rad50 (.(entry [rt$l_filename])<0, 16>, name [1]) + rad50 (.(entry [rt$l_filename]
;    0444  5                        )<16, 16>, name [4]);
;    0445  5                    type [0] = rad50 (.entry [rt$w_filetype], type [1]);
;    0446  5                    q_buffer [dsc$w_length] = 19;   ! Set up FAO output buffer
;    0447  5                    q_buffer [dsc$a_pointer] = .OutBuffer [Dsc$A_Pointer] + .OutBuffer [Dsc$W_Length];      !       [10]
;    0448  5                    $fao ($ascid ('!19<!AC.!AC!>'), q_buffer, q_buffer, name, type);        !              [10]
;    0449  5                    PutBuffer (.Count, OutBuffer, Q_Buffer) !                                              [10]
;    0450  5                    end
;    0451  5
;    0452  3                end;                                        !                                              [08]
;    0453  3
;    0454  3                If .ControlCFlag                            !                                              [09]
;    0455  3                Then                                        !                                              [09]
;    0456  4                    Begin                                   !                                              [09]
;    0457  4                    Dsr$DeAllocate_Pseudo_RPB (.RPB);       !                                              [09]
;    0458  4                    Return Rms$_Normal                      ! Abort gracefully on ^C                       [09]
;    0459  3                    End;                                    !                                              [09]
;    0460  3
;    0461  2            end;
;    0462  2
;    0463  2        if .OutBuffer [Dsc$W_Length] gtr 0 then $ds_printf ($Ascic ('!AS!/'), OutBuffer);   !              [10]
;    0464  2
;    0465  2        Dsr$DeAllocate_Pseudo_RPB (.RPB);
;    0466  2        rms$_normal
;    0467  1        end;                                                ! of rt11_direct


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20 79 72 6F 74 63 65 72 69 44 2F 21 2F 21 16  0007C P.AAK:   .ASCII  <22>\!/!/Directory _!AS!/!/\                    ;
                  2F 21 2F 21 53 41 21 5F     0008B                                                                 ;
      3E 21 43 41 21 2E 43 41 21 3C 39 31 21  00093 P.AAM:   .ASCII  \!19<!AC.!AC!>\                                ;
                                    0000000D  000A0 P.AAL:   .LONG   13                                             ;
                                    00000000' 000A4          .ADDRESS P.AAM                                         ;
                              2F 21 53 41 21 05 000A8 P.AAN:   .ASCII  <5>\!AS!/\                                   ;


                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                07FC 00000 RT11_DIRECT:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                 ; 0387
                                 5A 00000000G   9F  9E 00002    MOVAB   @#DS$PRINTF, R10                            ;
                                 59 00000000'   EF  9E 00009    MOVAB   P.AAK, R9                                   ;
                                 58     FE30    CF  9E 0000D    MOVAB   RAD50, R8                                   ;
                                 5E     F894    CE  9E 00015    MOVAB   -1132(SP), SP                               ;
                                 7E             01  7D 0001A    MOVQ    #1, -(SP)                                   ; 0405
                                                7E  7C 0001D    CLRQ    -(SP)                                       ;
                                                7E  D4 0001F    CLRL    -(SP)                                       ;
                                 7E       40    8F  9A 00021    MOVZBL  #64, -(SP)                                  ;
                   00000000G     EF             06  FB 00025    CALLS   #6, DSR$ALLOCATE_PSEUDO_RPB                 ;
                                 57             50  D0 0002C    MOVL    R0, RPB                                     ;
                                 56             01  D0 0002F    MOVL    #1, SEGMENT                                 ; 0406
                                          0C    AE  B4 00032    CLRW    OUTBUFFER                                   ; 0407
                              10 AE       1C    AE  9E 00035    MOVAB   T_BUFFER, OUTBUFFER+4                       ; 0408
                                          08    AC  DD 0003A    PUSHL   Q_DEVICE                                    ; 0409
```

```
                                                59  DD 0003D          PUSHL   R9                                                              ;
                                        6A      02  FB 0003F          CALLS   #2, DS$PRINTF                                                   ;
0050    8F              20      6E      00  2C 00042          MOVC5   #0, (SP), #32, #80, T_BUFFER                                    ; 0410
                                    1C  AE      00049
                                                56  D5 0004B 1$:      TSTL    SEGMENT                                                         ; 0418
                                                03  12 0004D          BNEQ    2$
                                              00A5  31 0004F          BRW     6$
                                                57  DD 00052 2$:      PUSHL   RPB                                                             ; 0425
                                        7E      21  7D 00054          MOVQ    #33, -(SP)                                                      ; 0420
                                7E      56      01  78 00057          ASHL    #1, SEGMENT, -(SP)                                              ; 0422
                                        6E      04  C0 0005B          ADDL2   #4, (SP)
                                        7E  0400    8F  3C 0005E      MOVZWL  #1024, -(SP)                                                    ; 0420
                                            0080    CE  9F 00063      PUSHAB  SEGBLK
                        00000000G       EF      06  FB 00067          CALLS   #6, BOO$QIO
                                        01      50  E8 0006E          BLBS    STATUS, 3$
                                                04  00071          RET
                                        56  6E  AE  3C 00072 3$:      MOVZWL  SEGBLK+2, SEGMENT                                               ; 0428
                                        54  72  AE  3C 00076          MOVZWL  SEGBLK+6, EXTRA_LENGTH                                          ; 0429
                                        50  68  AE  9E 0007A          MOVAB   SEGBLK-4, R0                                                    ; 0430
                                52      50      54  C3 0007E          SUBL3   EXTRA_LENGTH, R0, ADDRESS_FIRST
                                        52  0E A442  9E 00082 4$:      MOVAB   14(EXTRA_LENGTH)[ADDRESS_FIRST], -                              ; 0436
                                                                              ADDRESS_FIRST
                                        53      52  D0 00087          MOVL    ADDRESS_FIRST, R3
                                5F      63      0B  E0 0008A          BBS     #11, (R3), 5$                                                   ; 0438
                                F0      63      0A  E1 0008E          BBC     #10, (R3), 4$                                                   ; 0440
                                            15  AE  9F 00092          PUSHAB  NAME+1                                                          ; 0443
                                        7E  02  A3  3C 00095          MOVZWL  2(R3), -(SP)
                                        68      02  FB 00099          CALLS   #2, RAD50
                                        55      50  D0 0009C          MOVL    R0, R5
                                            18  AE  9F 0009F          PUSHAB  NAME+4                                                          ; 0444
                                        7E  04  A3  3C 000A2          MOVZWL  4(R3), -(SP)
                                        68      02  FB 000A6          CALLS   #2, RAD50
                14      AE              55      50  81 000A9          ADDB3   R0, R5, NAME
                                            01  AE  9F 000AE          PUSHAB  TYPE+1                                                          ; 0445
                                        7E  06  A3  3C 000B1          MOVZWL  6(R3), -(SP)
                                        68      02  FB 000B5          CALLS   #2, RAD50
                                        6E      50  90 000B8          MOVB    R0, TYPE
                                04      AE      13  B0 000BB          MOVW    #19, Q_BUFFER                                                   ; 0446
                                        50  0C  AE  3C 000BF          MOVZWL  OUTBUFFER, R0                                                   ; 0447
                                08      AE      10  BE40 9E 000C3     MOVAB   @OUTBUFFER+4[R0], Q_BUFFER+4
                                                5E  DD 000C9          PUSHL   SP                                                              ; 0448
                                            18  AE  9F 000CB          PUSHAB  NAME
                                            0C  AE  9F 000CE          PUSHAB  Q_BUFFER
                                            10  AE  9F 000D1          PUSHAB  Q_BUFFER
                                            24  A9  9F 000D4          PUSHAB  P.AAL
                    00000000G       00          05  FB 000D7          CALLS   #5, SYS$FAO
                                            04  AE  9F 000DE          PUSHAB  Q_BUFFER                                                        ; 0449
                                            10  AE  9F 000E1          PUSHAB  OUTBUFFER
                                            0C  AC  DD 000E4          PUSHL   COUNT
                                88      A8          03  FB 000E7          CALLS   #3, PUTBUFFER
                                                95  11 000EB          BRB     4$                                                              ; 0440
                                        11  00000000'   EF  E8 000ED 5$:      BLBS    CONTROLCFLAG, 7$                                        ; 0454
                                                FF54  31 000F4          BRW     1$
                                            0C  AF  B5 000F7 6$:      TSTW    OUTBUFFER                                                       ; 0463
                                                09  13 000FA          BEQL    7$
                                            0C  AE  9F 000FC          PUSHAB  OUTBUFFER
                                            2C  A9  9F 000FF          PUSHAB  P.AAN
```

```
                          6A            02  FB 00102          CALLS   #2, DS$PRINTF                                  ;
                                        57  DD 00105 7$:      PUSHL   RPB                                           ; 0465
                    00000000G   FF      01  FB 00107          CALLS   #1, DSR$DEALLOCATE_PSEUDO_RPB                 ;
                                50 00010001 8F  D0 0010E       MOVL    #65537, R0                                    ; 0467
                                            04 00115          RET                                                   ;

; Routine Size:  278 bytes,    Routine Base:  CODE + 0231
```

```
;    0468  1     %sbttl 'ANSI directory'
;    0469  1     routine ansi_direct (filespec, count) =
;    0470  1
;    0471  1     !++
;    0472  1     ! Functional description:
;    0473  1     !
;    0474  1     !       Search through a tape to find all files, and list them on the
;    0475  1     !       console.
;    0476  1     !
;    0477  1     !       If double tape marks are found instead of a file header the
;    0478  1     !       tape is rewound and the search continued until the first file found is
;    0479  1     !       encountered again.
;    0480  1     !
;    0481  1     ! Formal parameters:
;    0482  1     !       filespec        address of filespec block
;    0483  1     !       count           address of longword to get file count
;    0484  1     !
;    0485  1     ! Implicit inputs:
;    0486  1     !       none
;    0487  1     !
;    0488  1     ! Outputs:
;    0489  1     !       count           count of files
;    0490  1     !
;    0491  1     ! Implicit outputs:
;    0492  1     !       none
;    0493  1     !
;    0494  1     ! Routine value:
;    0495  1     !
;    0496  1     !       Standard RMS completion codes
;    0497  1     !
;    0498  1     ! Side effects:
;    0499  1     !
;    0500  1     !       The device and associated channel are accessed
;    0501  1     !
;    0502  1     !--
;    0503  1
;    0504  2         begin
;    0505  2
;    0506  2         external routine
;    0507  2             sys$search : addressing_mode (absolute),
;    0508  2             sys$open : addressing_mode (absolute);
;    0509  2
;    0510  2         bind routine
;    0511  2             next_file =
;    0512  2
;    0513  2                 if .dsa$v_user then sys$search else sys$open;
;    0514  2
;    0515  2         local
;    0516  2             filenam : bblock [nam$c_maxrss],
;    0517  2             filedesc : bblock [dsc$c_s_bln],
;    0518  2             res_len,
;    0519  2             status,
;    0520  2             fab : $fab_decl,                        ! FAB for open
;    0521  2             nam : $nam_decl,
;    0522  2             result : bblock [nam$c_maxrss],         ! Space for resultant string
;    0523  2             expand : bblock [nam$c_maxrss],         ! Space for expanded string
;    0524  2             first_file : bblock [132],              ! First file found
```

```
;    0525  2              first_len,
;    0526  2              buffer : vector [132, byte],          ! Space for output
;    0527  2              OutBuffer : BBlock [8];               ! Descriptor for output                    [10]
;    0528  2
;    0529  2          ch$fill (%c' ', 132, buffer);             ! Clear output buffer
;    0530  2          OutBuffer [Dsc$W_Length] = 0;             !                                          [10]
;    0531  2          OutBuffer [Dsc$A_Pointer] = Buffer;       !                                          [10]
;    0532  2          first_len = 132;
;    0533  2          ch$fill (0, 132, first_file);
;    0534  2          $ds_printf ($ascic ('!7!/Directory _!AS!/!/'), .filespec);
;    0535  2          filedesc [dsc$w_length] = nam$c_maxrss;
;    0536  2          filedesc [dsc$a_pointer] = filenam;
;    0537  2          $fao ($ascid ('!AS*.*;*'), filedesc, filedesc, .filespec);
;    0538  2          $fab_init (fab = fab, nam = nam, fop = rwo, fna = filenam, fns = .filedesc [dsc$w_length]);
;    0539  2          $nam_init (nam = nam, ess = nam$c_maxrss, rss = nam$c_maxrss, esa = expand, rsa = result);
;    0540  2
;    0541  2          if .dsa$v_user
;    0542  2          then
;    0543  2                                                    ! Init wildcard context
;    0544  3              begin
;    0545  3
;    0546  3              local
;    0547  3                  status;
;    0548  3
;    0549  3              status = $parse (fab = fab);
;    0550  3
;    0551  3              if not .status then return .status
;    0552  3
;    0553  2              end;
;    0554  2
;    0555  2          while (status = next_file (fab); fab [fab$v_rwo] = 0; .status) do
;    0556  3              begin
;    0557  3
;    0558  3              local
;    0559  3                  nam_len,
;    0560  3                  nam_ptr;
;    0561  3
;    0562  3              if not .dsa$v_user                    ! If we did an OPEN,
;    0563  3              then                                  !  we have to CLOSE, too.
;    0564  4                  begin
;    0565  4                  status = $close (fab = fab);
;    0566  4
;    0567  4                  if not .status then exitloop
;    0568  4
;    0569  3                  end;
;    0570  3
;    0571  3              res_len = .nam [nam$b_rsl];
;    0572  3
;    0573  3              if ch$compare (.first_len, first_file, .res_len, result, 0) eql 0 then exitloop;
;    0574  3
;    0575  3              if .(first_file)<0, 8> eql 0 then ch$move (first_len = .res_len, result, first_file);
;    0576  3
;    0577  3              nam_ptr = ch$find_ch (.res_len, result, %c':') + 1;     ! Look for end of 'MTAn:'
;    0578  3              nam_len = .res_len - (.nam_ptr - result);
;    0579  3              FileDesc [Dsc$W_Length] = .Nam_Len;
;    0580  3              FileDesc [Dsc$A_Pointer] = .Nam_Ptr;
;    0581  3              IF .WIDE_FLAG
```

```
;    0582  3              THEN Ch$Copy (.nam_len, .nam_ptr, %c' ', 132, buffer + .OutBuffer [Dsc$W_Length])          !       [17]
;    0583  3              ELSE Ch$Copy (.nam_len, .nam_ptr, %c' ', 19, buffer + .OutBuffer [Dsc$W_Length]);          !       [10]
;    0584  3          PutBuffer (.Count, OutBuffer, FileDesc);                                                              [10]
;    0585  3          If .ControlCFlag Then Return Rms$_Normal        ! Abort gracefully on ^C                              [08]
;    0586  3
;    0587  2          end;
;    0588  2
;    0589  2      if .OutBuffer [Dsc$W_Length] gtr 0 then $ds_printf ($Ascic ('!AS!/'), OutBuffer);   !                  [10]
;    0590  2
;    0591  2      if .status eql rms$_nmf then status = rms$_normal;
;    0592  2
;    0593  2      .status
;    0594  1      end;                                                ! of ansi_direct


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20 79 72 6F 74 63 65 72 69 44 2F 21 2F 21 16  000AE P.AAO:  .ASCII  <22>\!/!/Directory _!AS!/!/\                    ;
                        2F 21 2F 21 53 41 21 5F  000BD                                                              ;
                        2A 3B 2A 2E 2A 53 41 21  000C5 P.AAQ:  .ASCII  \!AS*.*;*\                                   ;
                                                 000CD         .BLKB   3                                            ;
                                    00000008     000D0 P.AAP:  .LONG   8                                            ;
                                    00000000'    000D4         .ADDRESS P.AAQ                                       ;
                              2F 21 53 41 21 05  000D8 P.AAR:  .ASCII  <5>\!AS!/\                                   ;

                                                                .EXTRN  SYS$SEARCH, SYS$OPEN
                                                                .EXTRN  SYS$PARSE, SYS$CLOSE

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                     0FFC 00000 ANSI_DIRECT:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0469
                                 5B 00000000'   EF   9E 00002  MOVAB   P.AAO, R11                                   ;
                                 5E      FB44   CE   9E 00009  MOVAB   -1212(SP), SP                                ;
      5A 0000FE03   9F           01             04   EF 0000E  EXTZV   #4, #1, @#^X0000FE03, R10                   ; 0513
                                 09             5A   E9 00017  BLBC    R10, 1$                                     ;
                                 58 00000000G   8F   D0 0001A  MOVL    #SYS$SEARCH, R8                             ;
                                                07   11 00021  BRB     2$                                          ;
                                 58 00000000G   8F   D0 00023  1$:     MOVL    #SYS$OPEN, R8                       ;
0084  8F                 20     6E             00   2C 0002A  2$:     MOVC5   #0, (SP), #32, #132, BUFFER         ; 0529
                                           08   AE      00031                                                      ;
                                 6E             B4   00033  CLRW    OUTBUFFER                                   ; 0530
                           04    AE        08   AE   9E 00035  MOVAB   BUFFER, OUTBUFFER+4                        ; 0531
                                 59        84   8F   9A 0003A  MOVZBL  #132, FIRST_LEN                            ; 0532
0084  8F                 00     6E             00   2C 0003E  MOVC5   #0, (SP), #0, #132, FIRST_FILE             ; 0533
                                       008C     CE      00045                                                      ;
                                          04    AC   DD 00048  PUSHL   FILESPEC                                    ; 0534
                                                5B   DD 0004B  PUSHL   R11                                         ;
                     00000000G   9F             02   FB 0004D  CALLS   #2, @#DS$PRINTF                           ;
                         FEFC    CD        FC   8F   9B 00054  MOVZBW  #252, FILEDESC                             ; 0535
                         FF00    CD      FF04   CD   9E 0005A  MOVAB   FILENAM, FILEDESC+4                        ; 0536
                                          04    AC   DD 00061  PUSHL   FILESPEC                                    ; 0537
                                       FEFC     CD   9F 00064  PUSHAB  FILEDESC                                    ;
                                       FEFC     CD   9F 00068  PUSHAB  FILEDESC                                    ;
                                          22    AB   9F 0006C  PUSHAB  P.AAP                                       ;
                     00000000G   00             04   FB 0006F  CALLS   #4, SYS$FAO                               ;
```

```
0050  8F  00  6E  FEAC  CD  00 2C 00076  MOVC5  #0, (SP), #0, #80, $RMS_PTR  ; 0538
                        FEAC  CD  0007D
                  FEAC  CD  5003  8F B0 00080  MOVW  #20483, $RMS_PTR
                  FEB0  CD    80  8F 9A 00087  MOVZBL  #128, $RMS_PTR+4
                  FEC2  CD        02 90 0008D  MOVB  #2, $RMS_PTR+22
                  FECB  CD        02 90 00092  MOVB  #2, $RMS_PTR+31
                  FED4  CD  FE4C  CD 9E 00097  MOVAB  NAM, $RMS_PTR+40
                  FED8  CD  FF04  CD 9E 0009E  MOVAB  FILENAM, $RMS_PTR+44
                  FEE0  CD  FEFC  CD 90 000A5  MOVB  FILEDESC, $RMS_PTR+52
0060  8F  00  6E        00 2C 000AC  MOVC5  #0, (SP), #0, #96, $RMS_PTR  ; 0539
                        FE4C  CD    000B3
                  FE4C  CD  6002  8F B0 000B6  MOVW  #24578, $RMS_PTR
                  FE4E  CD        04 8E 000BD  MNEGB  #4, $RMS_PTR+2
                  FE50  CD  020C  CE 9E 000C2  MOVAB  RESULT, $RMS_PTR+4
                  FE56  CD        04 8E 000C9  MNEGB  #4, $RMS_PTR+10
                  FE58  CD  0110  CE 9E 000CE  MOVAB  EXPAND, $RMS_PTR+12
                        0F        5A E9 000D5  BLBC  R10, 3$  ; 0541
                        FEAC  CD  9F 000D8  PUSHAB  FAB  ; 0549
              00000000G  00       01 FB 000DC  CALLS  #1, SYS$PARSE
                        01        50 E8 000E3  BLBS  STATUS, 3$  ; 0551
                                  04 000E6  RET
                        FEAC  CD  9F 000E7 3$:  PUSHAB  FAB  ; 0555
                        68        01 FB 000EB  CALLS  #1, (R8)
                        56        50 D0 000EE  MOVL  R0, STATUS
                  FEB0  CD    80  8F 8A 000F1  BICB2  #128, FAB+4
                        11        56 E9 000F7  BLBC  STATUS, 4$
                        14        5A E8 000FA  BLBS  R10, 5$  ; 0562
                        FEAC  CD  9F 000FD  PUSHAB  FAB  ; 0565
              00000000G  00       01 FB 00101  CALLS  #1, SYS$CLOSE
                        56        50 D0 00108  MOVL  R0, STATUS
                        03        56 E8 0010B 4$:  BLBS  STATUS, 5$  ; 0567
                              008D  31 0010E  BRW  12$
                        57  FE4F  CD  9A 00111 5$:  MOVZBL  NAM+3, RES_LEN  ; 0571
                        54        01 D0 00116  MOVL  #1, R4  ; 0573
      57        00  008C  CE        59 2D 00119  CMPC5  FIRST_LEN, FIRST_FILE, #0, RES_LEN, RESULT
                        020C  CE    00120
                              03 1A 00123  BGTRU  6$
                        54        01 D9 00125  SBWC  #1, R4
                              54  D5 00128 6$:  TSTL  R4
                              72  13 0012A  BEQL  12$
                        008C  CE  95 0012C  TSTB  FIRST_FILE  ; 0575
                              0B  12 00130  BNEQ  7$
                        59        57 D0 00132  MOVL  RES_LEN, FIRST_LEN
            008C  CE  020C  CE    57 28 00135  MOVC3  RES_LEN, RESULT, FIRST_FILE
            020C  CE    57        3A 3A 0013D 7$:  LOCC  #58, RES_LEN, RESULT  ; 0577
                              02  12 00143  BNEQ  8$
                              51  D4 00145  CLRL  R1
                              51  D6 00147 8$:  INCL  NAM_PTR
                        50  020C  CE 9E 00149  MOVAB  RESULT, R0  ; 0578
                        50        51 C2 0014E  SUBL2  NAM_PTR, R0
                  52    50        57 C1 00151  ADDL3  RES_LEN, R0, NAM_LEN
                  FEFC  CD        52 B0 00155  MOVW  NAM_LEN, FILEDESC  ; 0579
                  FF00  CD        51 D0 0015A  MOVL  NAM_PTR, FILEDESC+4  ; 0580
                        50        6E 3C 0015F  MOVZWL  OUTBUFFER, R0  ; 0582
                        0C  00000000'  EF E9 00162  BLBC  WIDE_FLAG, 9$  ; 0581
0084  8F  20  61        52 2C 00169  MOVC5  NAM_LEN, (NAM_PTR), #32, #132, BUFFER[R0]  ; 0582
                              08 AE40  00170
```

```
                                         08  11 00173       BRB     10$                                        ;
     13          20          61          52  2C 00175 9$:   MOVC5   NAM_LEN, (NAM_PTR), #32, #19, BUFFER[R0]   ; 0583
                                    08 AE40     0017A                                                          ;
                                   FEFC  CD  9F 0017D 10$:  PUSHAB  FILEDESC                                   ; 0584
                                     04  AE  9F 00181       PUSHAB  OUTBUFFER                                  ;
                                     08  AC  DD 00184       PUSHL   COUNT                                      ;
                        FB2D  CF         03  FB 00187       CALLS   #3, PUTBUFFER                              ;
                          03 00000000'   EF  E8 0018C       BLBS    CONTROLCFLAG, 11$                          ; 0585
                                       FF51  31 00193       BRW     3$                                         ;
                          50 00010001    8F  D0 00196 11$:  MOVL    #65537, R0                                 ;
                                             04 0019D       RET                                                ;
                                         6E  B5 0019E 12$:  TSTW    OUTBUFFER                                  ; 0589
                                         0C  13 001A0       BEQL    13$                                        ;
                                         5E  DD 001A2       PUSHL   SP                                         ;
                                     2A  AB  9F 001A4       PUSHAB  P.AAR                                      ;
                   00000000G  9F         02  FB 001A7       CALLS   #2, @#DS$PRINTF                            ;
                   000182CA   8F         56  D1 001AE 13$:  CMPL    STATUS, #99018                             ; 0591
                                         07  12 001B5       BNEQ    14$                                        ;
                          56 00010001    8F  D0 001B7       MOVL    #65537, STATUS                             ;
                          50             56  D0 001BE 14$:  MOVL    STATUS, R0                                 ; 0594
                                             04 001C1       RET                                                ;

; Routine Size:  450 bytes,    Routine Base:  CODE + 0347
```

```
;    0595  1     %sbttl 'ods directory'
;    0596  1     routine ods_direct (q_filspec, count) =
;    0597  2         begin
;    0598  2
;    0599  2         local
;    0600  2             status,
;    0601  2             fab : $fab_decl,                    ! FAB block to read directory file
;    0602  2             rab : $rab_decl,                    ! RAB block
;    0603  2             buffer : bblock [512],              ! Buffer to read record into
;    0604  2             t_directory : bblock [8],           ! Buffer for parsed directory
;    0605  2             q_directory : bblock [dsc$c_s_bln], ! Descriptor of parsed directory
;    0606  2             base_directory : bblock [dsc$c_s_bln], ! Descriptor of direct
;    0607  2             q_file : bblock [dsc$c_s_bln],      ! Descriptor for file name
;    0608  2             t_file : bblock [512],              ! Space for file name          [18]
;    0609  2             q_buffer : bblock [dsc$c_s_bln],    ! Descriptor for output buffer
;    0610  2             t_buffer : bblock [132];            ! Buffer to build file list    [18]
;    0611  2
;    0612  2         bind
;    0613  2             base_direct = $ascid ('0,0'),       ! Normal base directory
;    0614  2             q_direct = .q_filspec + dsc$c_s_bln : bblock;   ! Directory descriptor
;    0615  2
;    0616  2         local
;    0617  2             left,
;    0618  2             right,
;    0619  2             p_dot,
;    0620  2             l_dot,
;    0621  2             p_lpart,
;    0622  2             p_rpart,
;    0623  2             l_rpart,
;    0624  2             l_lpart,
;    0625  2             p_comma,
;    0626  2             l_rest;
;    0627  2
;    0628  2         ch$move (8, base_direct, base_directory);   ! Assume normal directory
;    0629  2         p_lpart = .q_direct [dsc$a_pointer] + 1;    ! Point past "["
;    0630  2         l_rest = .q_direct [dsc$w_length] - 2;      ! Exclude brackets
;    0631  2         p_comma = ch$find_ch (.l_rest, .p_lpart, %c',');    ! Find ',' if any
;    0632  2
;    0633  2         if .p_comma neq 0                           ! If there was a comma ("[octal,octal]")
;    0634  2         then
;    0635  3             begin
;    0636  3             p_rpart = .p_comma + 1;
;    0637  3             l_lpart = .p_comma - .p_lpart;          ! Length of left number
;    0638  3             l_rpart = .l_rest - .l_lpart - 1;       ! Length of right number
;    0639  3             left = scan$numeric (8, .l_lpart, .p_lpart);
;    0640  3             right = scan$numeric (8, .l_rpart, .p_rpart);
;    0641  3             q_directory [dsc$a_pointer] = t_directory;
;    0642  3             q_directory [dsc$w_length] = 8;
;    0643  4             $fao ($ascid ('!3O[!3O['), q_directory, q_directory, .left, .right)
;    0644  3             end
;    0645  2         else
;    0646  3             begin                                   ! Copy string, without '[]'
;    0647  3             q_directory [dsc$w_length] = .q_direct [dsc$w_length] - 2;
;    0648  3             q_directory [dsc$a_pointer] = .q_direct [dsc$a_pointer] + 1
;    0649  2             end;
;    0650  2
;    0651  2         q_file [dsc$w_length] = 512 ;               !                          [18]
```

```
;    0652  2          q_file [dsc$a_pointer] = t_file;
;    0653  2
;    0654  2      !
;    0655  2      ! Print out the header
;    0656  2      !
;    0657  2          $ds_printf ($ascic ('!/!/Directory _!AS[!AS]!/!/'), .q_filspec, q_directory);       ! Print header
;    0658  2      !
;    0659  2      ! Now, check for subdirectories, and kluge string around
;    0660  2      !
;    0661  2          p_dot = ch$find_ch (.l_rest, .p_lpart, %c'.');
;    0662  2
;    0663  2          if not ch$fail (.p_dot)                         ! If there are subdirectories
;    0664  3          then
;    0665  3              begin
;    0666  3
;    0667  3              local
;    0668  3                  last_dot;
;    0669  3
;    0670  3              while not ch$fail (.p_dot) do               ! Find last dot
;    0671  4                  begin
;    0672  4                  last_dot = .p_dot;
;    0673  4                  p_dot = ch$find_ch (.l_rest - (.p_dot - .p_lpart) - 1,
;    0674  3                      .p_dot + 1, %c'.')
;    0675  3                  end;
;    0676  3
;    0677  3              base_directory [dsc$w_length] = .last_dot - .p_lpart;
;    0678  3              base_directory [dsc$a_pointer] = .p_lpart;
;    0679  3              q_directory [dsc$w_length] = .l_rest - (.last_dot - .p_lpart) - 1;
;    0680  2              q_directory [dsc$a_pointer] = .last_dot + 1
;    0681  2              end;
;    0682  2
;    0683  2      !
;    0684  2      ! Now get the filename to look up
;    0685  2      !
;P   0686  2          $fao ($ascid ('!AS[!AS]!AS.DIR;0'),             ! Create file to look up (directory)
;    0687  2              q_file, q_file, .q_filspec, base_directory, q_directory);
;    0688  2          $fab_init (fab = fab, fna = .q_file [dsc$a_pointer], fns = .q_file [dsc$w_length], fac = get);
;    0689  2          $rab_init (rab = rab, fab = fab, ubf = buffer, usz = 512);
;    0690  2
;    0691  3          if not (status = $open (fab = fab))
;    0692  2          then
;    0693  3              return (selectone .status of
;    0694  3                      set
;    0695  3                      [rms$_fnf] : rms$_dnf;
;    0696  3                      [otherwise] : .status
;    0697  2                      tes);
;    0698  2
;    0699  3          if not (status = $connect (rab = rab))
;    0700  2          then
;    0701  3              begin
;    0702  3              $close (fab = fab);
;    0703  3              return .status
;    0704  2              end;
;    0705  2
;    0706  2          ch$fill (%c' ', 132, t_buffer);                 !       [18]
;    0707  2          q_buffer [dsc$w_length] = 0;
;    0708  2          q_buffer [dsc$a_pointer] = t_buffer;
```

```
;    0709  2          while (status = $get (rab = rab)) do
;    0710  3              Begin
;    0711  3
;    0712  3              case .fab [fab$b_rfm] from fab$c_fix to fab$c_var of
;    0713  3                  set
;    0714  3
;    0715  3                  [fab$c_fix] :
;    0716  3                      format_ods1 (.rab [rab$w_rsz], .rab [rab$l_rbf], q_buffer, .count);
;    0717  3
;    0718  3                  [fab$c_var] :
;    0719  3                      format_ods2 (.rab [rab$w_rsz], .rab [rab$l_rbf], q_buffer, .count);
;    0720  3
;    0721  3                  [outrange] :
;    0722  4                      begin
;    0723  4                      $close (fab = fab);
;    0724  4                      return ss$_filestruct           ! Bad file structure level
;    0725  3                      end;
;    0726  3                  tes;
;    0727  3
;    0728  3                  If .ControlCFlag Then Exitloop      ! Close file and leave on ^C                                [08]
;    0729  3
;    0730  2                  End;
;    0731  2
;    0732  2              if .q_buffer [dsc$w_length] neq 0 then $ds_printf ($ascic ('!AS!/'), q_buffer);
;    0733  2
;    0734  2              $close (fab = fab);
;    0735  3              (selectone .status of
;    0736  3                  set
;    0737  3                  [rms$_eof] : rms$_normal;
;    0738  3                  [otherwise] : .status
;    0739  3                  tes)
;    0740  1              end;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                                        30  2C  30  000DE P.AAT:  .ASCII  \0,0\                              ;
                                                                    000E1        .BLKB   3
                                                      00000003      000E4 P.AAS:  .LONG   3                                   ;
                                                      00000000'     000E8        .ADDRESS P.AAT                               ;
                                            4C  4F  33  21  4C  4F  33  21  000EC P.AAV:  .ASCII  \!3OL!3OL\                   ;
                                                      00000008      000F4 P.AAU:  .LONG   8                                   ;
                                                      00000000'     000F8        .ADDRESS P.AAV                               ;
20  79  72  6F  74  63  65  72  69  44  2F  21  2F  21  18  000FC P.AAW:  .ASCII  <27>\!/!/Directory _!AS[!AS]!/!/\           ;
        2F  21  2F  21  5D  53  41  21  5B  53  41  21  5F  0010B                                                             ;
52  49  44  2E  53  41  21  5D  53  41  21  5B  53  41  21  00118 P.AAY:  .ASCII  \!AS[!AS]!AS.DIR;0\                        ;
                                                    30  38  00127                                                           ;
                                                            00129        .BLKB   3
                                                      00000011      0012C P.AAX:  .LONG   17                                  ;
                                                      00000000'     00130        .ADDRESS P.AAY                               ;
                                        2F  21  53  41  21  05  00134 P.AAZ:  .ASCII  <5>\!AS!/\                           ;

                                                                BASE_DIRECT=        P.AAS
                                                                .EXTRN  SYS$CONNECT, SYS$GET

                                                                .PSECT  CODE,NOWRT,  SHR,2
```

```
                                        OFFC 00000 ODS_DIRECT:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0596
                                5B 00000000'  EF  9E 00002         MOVAB   BASE_DIRECT, R11             ;
                                5E      FAC0  CE  9E 00009         MOVAB   -1344(SP), SP                ;
                                50        04  AC  D0 0000E         MOVL    Q_FILSPEC, R0                ; 0614
                                56        08  A0  9E 00012         MOVAB   8(R0), R6                    ;
                          0294  CE            6B    08 28 00016    MOVC3   #8, BASE_DIRECT, BASE_DIRECTORY ; 0628
                          53        04        A6    01 C1 0001C    ADDL3   #1, 4(R6), R3                ; 0629
                                              58    53 D0 00021    MOVL    R3, P_LPART                  ;
                                              52    66 3C 00024    MOVZWL  (R6), R2                     ; 0630
                                              56    72 3E 00027    MOVAW   -(R2), L_REST                ;
                          68                  56    2C 3A 0002A    LOCC    #44, L_REST, (P_LPART)       ; 0631
                                              02    12 0002E       BNEQ    1$                           ;
                                              51    D4 00030       CLRL    R1                           ;
                                              51    D5 00032 1$:   TSTL    P_COMMA                      ; 0633
                                              55    13 00034       BEQL    2$                           ;
                                59        01  A1  9E 00036         MOVAB   1(R1), P_RPART               ; 0636
                          50                  51    58 C3 0003A    SUBL3   P_LPART, P_COMMA, L_LPART    ; 0637
                          54                  56    50 C3 0003E    SUBL3   L_LPART, L_REST, R4          ; 0638
                                57        FF  A4  9E 00042         MOVAB   -1(R4), L_RPART              ;
                                55            58    D0 00046       MOVL    P_LPART, R5                  ; 0639
                                54            50    D0 00049       MOVL    L_LPART, R4                  ;
                                53            08    D0 0004C       MOVL    #8, R3                       ;
                                   00000000G  EF  16 0004F         JSB     SCAN$NUMERIC                 ;
                                5A            50    D0 00055       MOVL    R0, LEFT                     ;
                                55            59    D0 00058       MOVL    P_RPART, R5                  ; 0640
                                54            57    D0 0005B       MOVL    L_RPART, R4                  ;
                                53            08    D0 0005E       MOVL    #8, R3                       ;
                                   00000000G  EF  16 00061         JSB     SCAN$NUMERIC                 ;
                          FD60  CD      FD64  CD  9E 00067         MOVAB   T_DIRECTORY, Q_DIRECTORY+4   ; 0641
                          029C  CE            08    B0 0006E       MOVW    #8, Q_DIRECTORY              ; 0642
                                              50    DD 00073       PUSHL   RIGHT                        ; 0643
                                              5A    DD 00075       PUSHL   LEFT                         ;
                                      FD5C    CD  9F 00077         PUSHAB  Q_DIRECTORY                  ;
                                      FD5C    CD  9F 0007B         PUSHAB  Q_DIRECTORY                  ;
                                        10    AB  9F 0007F         PUSHAB  P.AAU                        ;
                             00000000G  00            05    FB 00082    CALLS   #5, SYS$FAO              ;
                                                      0A    11 00089    BRB     3$                       ;
                                029C  CE            52    B0 0008B 2$:   MOVW    R2, Q_DIRECTORY          ; 0647
                                FD60  CD            53    D0 00090       MOVL    R3, Q_DIRECTORY+4        ; 0648
                                028C  CE      0200  8F    B0 00095 3$:   MOVW    #512, Q_FILE             ; 0651
                                0290  CE      008C  CE  9E 0009C         MOVAB   T_FILE, Q_FILE+4         ; 0652
                                      029C    CE  9F 000A3         PUSHAB  Q_DIRECTORY                  ; 0656
                                        04    AC  DD 000A7         PUSHL   Q_FILSPEC                    ;
                                        18    AB  9F 000AA         PUSHAB  P.AAW                        ;
                             00000000G  9F            03    FB 000AD    CALLS   #3, @#DS$PRINTF          ;
                          68                  56    2E 3A 000B4    LOCC    #46, L_REST, (P_LPART)       ; 0660
                                              02    12 000B8       BNEQ    4$                           ;
                                              51    D4 000BA       CLRL    R1                           ;
                                              51    D5 000BC 4$:   TSTL    P_DOT                        ; 0662
                                              39    13 000BE       BEQL    7$                           ;
                                              51    D5 000C0 5$:   TSTL    P_DOT                        ; 0669
                                              17    13 000C2       BEQL    6$                           ;
                                52            51    D0 000C4       MOVL    P_DOT, LAST_DOT              ; 0671
                          55            58    51    C3 000C7       SUBL3   P_DOT, P_LPART, R5           ; 0672
                                        50      FF  A546  9E 000CB         MOVAB   -1(R5)[L_REST], R0   ;
```

```
                    01    A1              50              2E  3A 000D0          LOCC    #46, R0, 1(P_DOT)                       ;
                                                        E9  12 000D5          BNEQ    5$                                      ;
                                                        51  D4 000D7          CLRL    R1                                      ;
                                                        E5  11 000D9          BRB     5$                                      ;
                          50              52            58  C3 000DB 6$:      SUBL3   P_LPART, LAST_DOT, R0                   ; 0676
                              0294        CE            50  B0 000DF          MOVW    R0, BASE_DIRECTORY                      ;
                              0298        CE            58  D0 000E4          MOVL    P_LPART, BASE_DIRECTORY+4               ; 0677
                          54              56            50  C3 000E9          SUBL3   R0, L_REST, R4                          ; 0678
                    029C  CE              54            01  A3 000ED          SUBW3   #1, R4, Q_DIRECTORY                     ;
                              FD60        CD      01    A2  9E 000F3          MOVAB   1(R2), Q_DIRECTORY+4                    ; 0679
                                          029C    CE    9F  9E 000F9 7$:      PUSHAB  Q_DIRECTORY                             ; 0686
                                          0298    CE    9F  9E 000FD          PUSHAB  BASE_DIRECTORY                          ;
                                            04    AC    DD  9E 00101          PUSHL   Q_FILSPEC                               ;
                                          0298    CE    9F  9E 00104          PUSHAB  Q_FILE                                  ;
                                          029C    CE    9F  9E 00108          PUSHAB  Q_FILE                                  ;
                                            48    AB    9F  9E 0010C          PUSHAB  P.AAX                                   ;
                    00000000G             00            06  FB 0010F          CALLS   #6, SYS$FAO                             ;
    0050    8F              00              6E            00  2C 00116          MOVC5   #0, (SP), #0, #80, $RMS_PTR             ; 0687
                                                    B0    AD      0011D
                                  B0    AD      5003    8F  B0 0011F          MOVW    #20483, $RMS_PTR                        ;
                                  C6    AD              02  90 00125          MOVB    #2, $RMS_PTR+22                         ;
                                  CF    AD              02  90 00129          MOVB    #2, $RMS_PTR+31                         ;
                                  DC    AD      0290    CE  D0 0012D          MOVL    Q_FILE+4, $RMS_PTR+44                   ;
                                  E4    AD      028C    CE  90 00133          MOVB    Q_FILE, $RMS_PTR+52                     ;
    0044    8F              00              6E            00  2C 00139          MOVC5   #0, (SP), #0, #68, $RMS_PTR             ; 0688
                                                    FF6C  CD      00140
                                FF6C    CD      4401    8F  B0 00143          MOVW    #17409, $RMS_PTR                        ;
                                  8C    AD      0200    8F  B0 0014A          MOVW    #512, $RMS_PTR+32                       ;
                                  90    AD      FD6C    CD  9E 00150          MOVAB   BUFFER, $RMS_PTR+36                     ;
                                  A8    AD        B0    AD  9E 00156          MOVAB   FAB, $RMS_PTR+60                        ;
                                                    B0    AD  9F 0015B          PUSHAB  FAB                                     ; 0690
                    00000000G     00              01    FB  9E 0015E          CALLS   #1, SYS$OPEN                            ;
                                  56              50    D0  00165             MOVL    R0, STATUS                              ;
                                  11              56    E8  00168             BLBS    STATUS, 8$                              ;
                    00018292      8F              56    D1  0016B             CMPL    STATUS, #98962                          ; 0694
                                                  23    12  00172             BNEQ    9$                                      ;
                                  50  0001C04A     3F    D0  00174             MOVL    #114762, R0                             ;
                                                        04  0017B             RET                                             ;
                                        FF6C      CD    9F  0017C 8$:         PUSHAB  RAB                                     ; 0698
                    00000000G     00              01    FB  00180             CALLS   #1, SYS$CONNECT                         ;
                                  56              50    D0  00187             MOVL    R0, STATUS                              ;
                                  0D              56    E8  0018A             BLBS    STATUS, 10$                             ;
                                                    B0  AD  9F  0018D         PUSHAB  FAB                                     ; 0701
                    00000000G     00              01    FB  00190             CALLS   #1, SYS$CLOSE                           ;
                                                  0099  31  00197 9$:         BRW     18$                                     ; 0702
    0084    8F              20              6E            00  2C 0019A 10$:      MOVC5   #0, (SP), #32, #132, T_BUFFER           ; 0705
                                                          6E      001A1
                                          0084    CE    B4  001A2             CLRW    Q_BUFFER                                ; 0706
                                    0088  CE            6E  9E 001A6          MOVAB   T_BUFFER, Q_BUFFER+4                    ; 0707
                                        FF6C    CD      9F  001AB 11$:        PUSHAB  RAB                                     ; 0709
                    00000000G     00              01    FB  001AF             CALLS   #1, SYS$GET                             ;
                                  56              50    D0  001B6             MOVL    R0, STATUS                              ;
                                  48              56    E9  001B9             BLBC    STATUS, 16$                             ;
                    01              01    CF      AD  8F  001BC             CASEB   FAB+31, #1, #1                          ; 0712
                              0029          0014        001C1 12$:          .WORD   13$-12$,-                               ;
                                                                                    14$-12$                                 ;
                                                    B0  AD  9F  001C5         PUSHAB  FAB                                     ; 0723
```

```
                    00000000G  00              01  FB 001C8          CALLS   #1, SYS$CLOSE                                    ;
                               50      08C0     8F  3C 001CF          MOVZWL  #2240, R0                                       ; 0724
                                               04  001D4              RET                                                     ;
                                         08     AC  DD 001D5 13$:     PUSHL   COUNT                                           ; 0716
                                       0088     CE  9F 001D8          PUSHAB  Q_BUFFER                                        ;
                                         94     AD  DD 001DC          PUSHL   RAB+40                                          ;
                                  7E     8E     AD  3C 001DF          MOVZWL  RAB+34, -(SP)                                   ;
                          F9DA    CF             04  FB 001E3          CALLS   #4, FORMAT_ODS1                                 ;
                                               13  11 001E8          BRB     15$                                             ;
                                         08     AC  DD 001EA 14$:     PUSHL   COUNT                                           ; 0719
                                       0088     CE  9F 001ED          PUSHAB  Q_BUFFER                                        ;
                                         94     AD  DD 001F1          PUSHL   RAB+40                                          ;
                                  7E     8E     AD  3C 001F4          MOVZWL  RAB+34, -(SP)                                   ;
                          FA64    CF             04  FB 001F8          CALLS   #4, FORMAT_ODS2                                 ;
                                  A7 00000000'   EF  E9 001FD 15$:     BLBC    CONTROLCFLAG, 11$                               ; 0728
                                       0084     CE  B5 00204 16$:     TSTW    Q_BUFFER                                        ; 0732
                                               0E  13 00208          BEQL    17$                                             ;
                                       0084     CE  9F 0020A          PUSHAB  Q_BUFFER                                        ;
                                         50     AB  9F 0020E          PUSHAB  P.AAZ                                           ;
                    00000000G  9F                04  FB 00211          CALLS   #2, @#DS$PRINTF                                 ;
                                         B0     AD  9F 00218 17$:     PUSHAB  FAB                                             ; 0734
                    00000000G  00                01  FB 0021B          CALLS   #1, SYS$CLOSE                                   ;
                    0001827A   8F                56  D1 00222          CMPL    STATUS, #98938                                  ; 0737
                                               08  12 00229          BNEQ    18$                                             ;
                                  50 00010001  8F  D0 0022B          MOVL    #65537, R0                                      ;
                                               04  00232              RET                                                     ;
                                  50           56  D0 00233 18$:     MOVL    STATUS, R0                                      ; 0738
                                               04  00236              RET                                                     ; 0740

; Routine Size:  567 bytes,    Routine Base:  CODE + 0509
```

```
;    0741  1      %sbttl 'Main code'
;    0742  1
;    0743  1      global routine dsx$directory (cli_block) : call_cli novalue =
;    0744  1
;    0745  1      !++
;    0746  1      !
;    0747  1      ! Functional description:
;    0748  1      !       This code just lists the file names and versions in a directory
;    0749  1      !
;    0750  1      ! Inputs:
;    0751  1      !       cli_block               pointer to cli block
;    0752  1      !
;    0753  1      ! Implicit inputs:
;    0754  1      !       none
;    0755  1      !
;    0756  1      ! Outputs:
;    0757  1      !       none
;    0758  1      !
;    0759  1      ! Implicit outputs:
;    0760  1      !       types directory of device
;    0761  1      !
;    0762  1      ! Side effects:
;    0763  1      !       accesses device via RMS
;    0764  1      !
;    0765  1      ! Return status:
;    0766  1      !       none
;    0767  1      !--
;    0768  1
;    0769  2          begin
;    0770  2
;    0771  2          Routine InterceptControlC =
;    0772  3              Begin
;    0773  4              (ControlCFlag = 1)      ! Just set the flag
;    0774  2              End;

                                        0000 00000 INTERCEPTCONTROLC:
                                                   .WORD   Save nothing                          ; 0771
                    00000000'  EF       01  90 00002       MOVB    #1, CONTROLCFLAG             ; 0773
                               50       01  D0 00009       MOVL    #1, R0                       ; 0774
                                            04 0000C       RET                                  ;

; Routine Size:  13 bytes,    Routine Base:  CODE + 0740

;    0775  2
;    0776  2          local
;    0777  2              count,
;    0778  2              length,
;    0779  2              pointer,
;    0780  2              status,                 !                                            [05]
;    0781  2              unit,
;    0782  2              devlen,
;    0783  2              devptr,
```

```
;  0784  2          FileMatch : Vector [Nam$C_MaxRss, Byte],        !                                                  [10]
;  0785  2          q_fileblock : bblock [dsc$c_s_bln*5];   ! File part descriptors
;  0786  2
;  0787  2      map
;  0788  2          cli_block : ref bblock;
;  0789  2
;  0790  2      Bind                                        !                                                          [10]
;  0791  2          WildCardMask = $Ascid ('*.*;*') : Block [, Byte],       !                                      [10]
;  0792  2
;  0793  2      ControlCFlag = 0;                           ! Initially clear the flag                             [08]
;  0794  2      $Ds_CntrlC (AstAdr = InterceptControlC);    ! Set Control C handler
;  0795  2      breakup_file (def$c_bkstp_len, def$q_backstop, q_fileblock);        ! Merge last-chance spec
;  0796  2      breakup_file (.def$q_dev [dsc$w_length], .def$q_dev [dsc$a_pointer], q_fileblock);
;  0797  2                                                  ! Merge default device
;  0798  2      breakup_file (.def$q_dir [dsc$w_length], .def$q_dir [dsc$a_pointer], q_fileblock);
;  0799  2                                                  ! Merge default directory
;  0800  2      BreakUp_File (.WildCardMask [Dsc$W_Length],  WildCardMask [Dsc$A_Pointer], Q_FileBlock);    !      [10]
;  0801  2      breakup_file (.cli_block [cli$w_file_len], .cli_block [cli$l_file_adr], q_fileblock);
;  0802  2                                                  ! Merge operator file spec
;  0803  2      FileSpecMatch [Dsc$W_Length] = 0;           ! Initialize descriptor                                [10]
;  0804  2      FileSpecMatch [Dsc$A_Pointer] = FileMatch;  !                                                      [10]
;  0805  2
;  0806  2      Incr Index From 2 to 4 Do                   ! Merge the final spec                                 [10]
;  0807  3          Begin                                   !                                                      [10]
;  0808  3
;  0809  3          Bind                                    !                                                      [10]
;  0810  3              Desc = Q_FileBlock + .Index*8 : Block [, Byte];     !                                      [10]
;  0811  3
;  0812  3          If .Desc [Dsc$W_Length] + .FileSpecMatch [Dsc$W_Length] Gtr Nam$C_MaxRss       !              [10]
;  0813  3          Then                                    !                                                      [10]
;  0814  3              Exitloop;                           !                                                      [10]
;  0815  3
;  0816  3          If .Desc [Dsc$W_Length] Neq 0           !                                                      [10]
;  0817  3           And .Desc [Dsc$A_Pointer] Neq 0        !                                                      [10]
;  0818  3          Then                                    !                                                      [10]
;  0819  4              Begin                               !                                                      [10]
;  0820  4
;  0821  4              Bind                                !                                                      [10]
;  0822  4                  Destination = .FileSpecMatch [Dsc$W_Length] + .FileSpecMatch [Dsc$A_Pointer];   !      [10]
;  0823  4
;  0824  4              Ch$Move (                           !                                                      [10]
;  0825  4                  .Desc [Dsc$W_Length],           !                                                      [10]
;  0826  4                  .Desc [Dsc$A_Pointer],          !                                                      [10]
;  0827  4                  Destination);                   !                                                      [10]
;  0828  4
;  0829  4              If .Index Eql 4                     ! Make sure Version starts with ';', not '.'           [10]
;  0830  4              Then                                !                                                      [10]
;  0831  4                  Ch$WChar (%C';', Destination);  !                                                      [10]
;  0832  4
;  0833  4              FileSpecMatch [Dsc$W_Length] = .FileSpecMatch [Dsc$W_Length] + .Desc [Dsc$W_Length];      !  [10]
;  0834  4              End                                 !                                                      [10]
;  0835  4
;  0836  2          End;
;  0837  2
;  0838  2      If Ch$Eql (                                 ! Don't bother matching if it's all wildcarded         [10]
;  0839  2          .FileSpecMatch [Dsc$W_Length],          !                                                      [10]
;  0840  2          .FileSpecMatch [Dsc$A_Pointer],         !                                                      [10]
```

```
;    0841  2              .WildCardMask [Dsc$W_Length],                                                                  [10]
;    0842  2              .WildCardMask [Dsc$A_Pointer],                                                                 [10]
;    0843  2              0)                                  (0 fill)                                                   [10]
;    0844  2          Then                                                                                               [10]
;    0845  2              FileSpecMatch [Dsc$W_Length] = 0;                                                              [10]
;    0846  2
;    0847  3          begin
;    0848  3
;    0849  3          global register
;    0850  3              zero = 0,
;    0851  3              one = 1;
;    0852  3
;    0853  3          linkage
;    0854  3              jsb_device = jsb (register = 4, register = 5) : global (zero = 0, one = 1)
;    0855  3              nopreserve (2, 3);
;    0856  3
;    0857  3          external routine
;    0858  3              scan$device : jsb_device novalue addressing_mode (long_relative);
;    0859  3
;    0860  3          scan$device (.q_fileblock [dsc$w_length], .q_fileblock [dsc$a_pointer]);
;    0861  3                                                  ! Decode/check the device name
;    0862  3          devlen = .zero<0, 16>;                  ! Pull out length of device
;    0863  3          unit = .zero<16, 16, 1>;                ! Get unit number, sign-extended
;    0864  3          devptr = .one                           ! Point to device name
;    0865  2          end;
;    0866  2
;    0867  2          count = 0;                              ! Initialize file count
;    0868  3          status = (                  ! Do directory & print message if error                             [05]
;    0869  3
;    0870  3              if .(.devptr)<0, 24> eql %ascii'CSA'    ! If console
;    0871  3              then
;    0872  4                  (if .unit lss 0 then 0 else If .Dsa$V_User                                                 [16]
;    0873  5                                              Then Begin                                                     [16]
;    0874  5                                                   $Ds_Printf ($Ascic ('!/%DS-F-USER, can''t access console device in u
;    0875  5                                                   Return 0                                               [16]
;    0876  5                                                   End                                                    [16]
;    0877  4                                              ELSE if .DSA$GL_SID[PR$V_SID_TYPE] EQL PR$_SID_TYPZZZ      !       [16]
;    0878  4                                                   THEN ODS_DIRECT (Q_FILEBLOCK, COUNT)                !       [16]
;    0879  4                                                   ELSE rti1_direct (.unit, q_fileblock, count)),
;    0880  3              else
;    0881  4                  begin
;    0882  4
;    0883  4                  local
;    0884  4                      ptable : ref bblock field ($ds_hpo_f);
;    0885  4
;    0886  4
;    0887  4                  loc$ptable (.devlen, .devptr, -1, ptable);  !                                          [07]
;    0888  4
;    0889  5                  (if .ptable eql 0 then 0 else selectone .(ptable [hp$t_type] + 1)<0, 16> of
;    0890  5                                              ! check device type
;    0891  5                      set
;    0892  5                      [%ascii'RP', %ascii'RM', %ascii'RK', %ascii'DI',%ascii'RC'                  !       [13]
;    0893  5                          %ascii'R8', %Ascii'RA', %ascii'RL'] : ods_direct (q_fileblock, count);  !       [11]
;    0894  5                      [%ascii'TA',%ascii'TE', %Ascii'TU', %ascii'TS'] :
;    0895  5                          ansi_direct (q_fileblock, count);                                       !       [15]
;    0896  5                      [otherwise] : rms$_dev
;    0897  5                      tes)
```

```
;    0898  4                end
;    0899  4
;    0900  2            );
;    0901  2
;    0902  2            if .status eql 0                !                                                                   [05]
;    0903  2            then                            !                                                                   [05]
; P  0904  2                $ds_printf ($ascic ('!/%DS-F-DEVNOTATCH, Device !AD not attached!/'),   !                       [05]
;    0905  3                    .devlen,  .devptr)      !                                                                   [05]
;    0906  2            else                            !                                                                   [05]
;    0907  2                dsr$completion (.status);   !                                                                   [05]
;    0908  2
;    0909  2            If .ControlCFlag Then Return;   ! No final printout on ^C                                                  [08]
;    0910  2
;    0911  2            $Ds_CntrlC ();                  ! Return ^C control to kernel                                       [08]
;    0912  2
;    0913  2            if .status                      !                                                                   [05]
;    0914  2            then                            !                                                                   [05]
;    0915  3                $ds_printf ($ascic ('!/Total of !ZL files.!/'), .count) !                                       [05]
;    0916  3
;    0917  1            end;                                        ! Of dsx$directory


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                    2A 3B 2A 2E 2A  0013A P.ABB:  .ASCII  \*.*;*\                                           ;
                                                    0013F         .BLKB   1                                                 ;
                                          00000005  00140 P.ABA:  .LONG   5                                                 ;
                                          00000000' 00144         .ADDRESS P.ABB                                            ;
20 2C 52 45 53 55 2D 46 2D 53 44 25 2F 21 39    00148 P.ABC:  .ASCII  \9!/%DS-F-USER, can't access console devi\           ;
6F 63 20 73 73 65 63 63 61 20 74 27 6E 61 63    00157                                                                     ;
                  69 76 65 64 20 65 6C 6F 73 6E 00166                                                                     ;
65 64 6F 6D 20 72 65 73 75 20 6E 69 20 65 63    00170         .ASCII  \ce in user mode.!/\                                ;
                                 2F 21 2E       0017F                                                                     ;
54 4F 4C 56 45 44 2D 46 2D 53 44 25 2F 21 2D    00182 P.ABD:  .ASCII  \-!/%DS-F-DEVNOTATCH, Device !AD not atta\           ;
41 21 20 65 63 69 76 65 44 20 2C 48 43 54 41    00191                                                                     ;
               61 74 74 61 20 74 6F 6E 20 44    001A0                                                                     ;
                           2F 21 64 65 68 63    001AA         .ASCII  \ched!/\                                            ;
4C 5A 21 20 66 6F 20 6C 61 74 6F 54 2F 21 17    001B0 P.ABE:  .ASCII  <23>\!/Total of !ZL files.!/\                       ;
                  2F 21 2E 73 65 6C 69 66 20    001BF                                                                     ;

                                                WILDCARDMASK::      P.ABA
                                                        .EXTRN  DS$CNTRLC, SCAN$DEVICE

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                          0FFC  00000           .ENTRY  DSX$DIRECTORY, Save R2,R3,R4,R5,R6,R7,R8,-          ; 0743
                                                                R9,R10,R11                                                  ;
                                    5B      EE    AF  9E  00002         MOVAB   INTERCEPTCONTROLC, R11                      ;
                                    5A 00000000'  EF  9E  0C006         MOVAB   WILDCARDMASK+4, R10                         ;
                                    59 00000000'  EF  9E  0C00D         MOVAB   FILESPECMATCH, R9                           ;
                                    5E    FED8    CE  9E  00014         MOVAB   -296(SP), SP                                ;
                                          08      A9  94  00019         CLRB    CONTROLCFLAG                                ; 0794
                                                  7E  D4  0001C         CLRL    -(SP)                                       ; 0794
                                                  5B  DD  0001E         PUSHL   R11                                         ;
                                00000000G 9F      02  FB  00020         CALLS   #2, @#DS$CNTRLC                             ;
                                          04      AE  9F  00027         PUSHAB  Q_FILEBLOCK                                 ; 0795
```

```
                                000000000G   EF   9F 0002A          PUSHAB  DEF$Q_BACKSTOP                                  ;
                                000000000G   8F   DD 00030          PUSHL   #DEF$C_BKSTP_LEN                                ;
                          0000G CF                03 FB 00036          CALLS   #3, BREAKUP_FILE                                ;
                                         04     AE 9F 0003B          PUSHAB  Q_FILEBLOCK                                     ; 0796
                                000000000G   EF   DD 0003E          PUSHL   DEF$Q_DEV+4                                     ;
                             7E 00C000000G   EF   3C 00044          MOVZWL  DEF$Q_DEV, -(SP)                                ;
                          0000G CF                03 FB 0004B          CALLS   #3, BREAKUP_FILE                                ;
                                         04     AE 9F 00050          PUSHAB  Q_FILEBLOCK                                     ; 0798
                                000000000G   EF   DD 00053          PUSHL   DEF$Q_DIR+4                                     ;
                             7E 000000000G   EF   3C 00059          MOVZWL  DEF$Q_DIR, -(SP)                                ;
                          0000G CF                03 FB 00060          CALLS   #3, BREAKUP_FILE                                ;
                                         04     AE 9F 00065          PUSHAB  Q_FILEBLOCK                                     ; 0800
                                         6A     DD 00068          PUSHL   WILDCARDMASK+4                                  ;
                             7E          FC     AA 3C 0006A          MOVZWL  WILDCARDMASK, -(SP)                             ;
                          0000G CF                03 FB 0006E          CALLS   #3, BREAKUP_FILE                                ;
                                         04     AE 9F 00073          PUSHAB  Q_FILEBLOCK                                     ; 0801
                                         0C     A2 DD 00076          PUSHL   12(CLI_BLOCK)                                   ;
                             7E          08     A2 3C 00079          MOVZWL  8(CLI_BLOCK), -(SP)                             ;
                          0000G CF                03 FB 0007D          CALLS   #3, BREAKUP_FILE                                ;
                                                 69 B4 00082          CLRW    FILESPECMATCH                                   ; 0803
                            04   A9          2C   AE 9E 00084          MOVAB   FILEMATCH, FILESPECMATCH+4                      ; 0804
                                 57                02 D0 00089          MOVL    #2, INDEX                                       ; 0806
                                 56          04 AE47 7E 0008C 1$:      MOVAQ   Q_FILEBLOCK[INDEX], R6                          ; 0810
                                 50                66 3C 00091          MOVZWL  (R6), R0                                        ; 0812
                                 51                69 3C 00094          MOVZWL  FILESPECMATCH, R1                               ;
                                 50                51 C0 00097          ADDL2   R1, R0                                          ;
                       000000FC  8F                50 D1 0009A          CMPL    R0, #252                                        ;
                                                   24 14 000A1          BGTR    4$                                              ;
                                                   66 B5 000A3          TSTW    (R6)                                            ; 0816
                                                   1C 13 000A5          BEQL    3$                                              ;
                                          04       A6 D5 000A7          TSTL    4(R6)                                           ; 0817
                                                   17 13 000AA          BEQL    3$                                              ;
                                      58           69 3C 000AC          MOVZWL  FILESPECMATCH, R8                               ; 0822
                                      58     04    A9 C0 000AF          ADDL2   FILESPECMATCH+4, R8                             ;
                                68          04    B6 66 28 000B3          MOVC3   (R6), @4(R6), (R8)                              ; 0824
                                      04           57 D1 000B8          CMPL    INDEX, #4                                       ; 0829
                                                   03 12 000BB          BNEQ    2$                                              ;
                                      68           3B 90 000BD          MOVB    #59, (R8)                                       ; 0831
                                      69           66 A0 000C0 2$:      ADDW2   (R6), FILESPECMATCH                             ; 0833
                                C5                57 04 F3 000C3 3$:    AOBLEQ  #4, INDEX, 1$                                   ; 0816
                          FC    AA    00          04 B9 69 2D 000C7 4$:    CMPC5   FILESPECMATCH, @FILESPECMATCH+4, #0, -          ; 0838
                                                      00 BA    000CE             WILDCARDMASK, @WILDCARDMASK+4                   ;
                                                   02 12 000D0          BNEQ    5$                                              ;
                                                   69 B4 000D2          CLRW    FILESPECMATCH                                   ; 0845
                                      55     08    AE D0 000D4 5$:       MOVL    Q_FILEBLOCK+4, R5                               ; 0860
                                      54     04    AE 3C 000D8          MOVZWL  Q_FILEBLOCK, R4                                 ;
                                         000000000G   EF 16 000DC          JSB     SCAN$DEVICE                                     ;
                                      54           50 3C 000E2          MOVZWL  ZERO, DEVLEN                                    ; 0862
                                52                50 F0 8F 78 000E5          ASHL    #-16, ZERO, UNIT                                ; 0863
                                      53           51 D0 000EA          MOVL    ONE, DEVPTR                                     ; 0864
                                                   6E D4 000ED          CLRL    COUNT                                           ; 0867
                00415343  8F      63             18 00 ED 000EF          CMPZV   #0, #24, (DEVPTR), #4281155                     ; 0870
                                                   2E 12 000F8          BNEQ    7$                                              ;
                                                   52 D5 000FA          TSTL    UNIT                                            ; 0872
                                                   3D 19 000FC          BLSS    8$                                              ;
                                0B 0000FE03        9F 04 E1 000FE          BBC     #4, @#^X0000FE03, 6$                            ;
                                                      04 AA 9F 00106       PUSHAB  P.ABC                                           ; 0874
```

```
                00000000G  9F              01  FB 00109          CALLS   #1, @#DS$PRINTF                          ;
                                           04  00110          RET                                              ; 0875
                           05 0000FE17      9F  91 00111 6$:   CMPB    @#^X0000FE17, #5                         ; 0877
                                            6i  13 00118       BEQL    10$                                      ;
                                            5E  DD 0011A       PUSHL   SP                                       ; 0879
                                       08   AE  9F 0011C       PUSHAB  Q_FILEBLOCK                              ;
                                            52  DD 0011F       PUSHL   UNIT                                     ;
                      FAF1  CB              03  FB 00121       CALLS   #3, RT11_DIRECT                          ;
                                            5D  11 00126       BRB     11$                                      ;
                            52              01  CE 00128 7$:   MNEGL   #1, R2                                  ; 0887
                            51              53  D0 0012B       MOVL    DEVPTR, R1                               ;
                            50              54  D0 0012E       MOVL    DEVLEN, R0                               ;
                                00000000G   EF  16 00131       JSB     LOC$PTABLE                               ;
                                            51  D5 00137       TSTL    PTABLE                                   ; 0889
                                            04  12 00139       BNEQ    9$                                       ;
                                            52  D4 0013B 8$:   CLRL    STATUS                                   ;
                                            7A  11 0013D       BRB     16$                                      ;
                            50       27     A1  3C 0013F 9$:   MOVZWL  39(PTABLE), R0                           ;
                      3852  8F              50  B1 00143       CMPW    R0, #14418                               ; 0892
                                            31  13 00148       BEQL    10$                                      ;
                      4152  8F              50  B1 0014A       CMPW    R0, #16722                               ;
                                            2A  13 0014F       BEQL    10$                                      ;
                      4352  8F              50  B1 00151       CMPW    R0, #17234                               ;
                                            23  13 00156       BEQL    10$                                      ;
                      4944  8F              50  B1 00158       CMPW    R0, #18756                               ;
                                            1C  13 0015D       BEQL    10$                                      ;
                      4B52  8F              50  B1 0015F       CMPW    R0, #19282                               ;
                                            15  13 00164       BEQL    10$                                      ;
                      4C52  8F              50  B1 00166       CMPW    R0, #19538                               ;
                                            0E  13 0016B       BEQL    10$                                      ;
                      4D52  8F              50  B1 0016D       CMPW    R0, #19794                               ;
                                            07  13 00172       BEQL    10$                                      ;
                      5052  8F              50  B1 00174       CMPW    R0, #20562                               ;
                                            0C  12 00179       BNEQ    12$                                      ;
                                            5E  DD 0017B 10$:   PUSHL   SP                                       ; 0893
                                       08   AE  9F 0017D       PUSHAB  Q_FILEBLOCK                              ;
                      FDC9  CB              02  FB 00180       CALLS   #2, ODS_DIRECT                           ;
                                            2F  11 00185 11$:   BRB     15$                                      ;
                      4154  8F              50  B1 00187 12$:   CMPW    R0, #16724                               ; 0894
                                            15  13 0018C       BEQL    13$                                      ;
                      4554  8F              50  B1 0018E       CMPW    R0, #17748                               ;
                                            0E  13 00193       BEQL    13$                                      ;
                      5354  8F              50  B1 00195       CMPW    R0, #21332                               ;
                                            07  13 0019A       BEQL    13$                                      ;
                      5554  8F              50  B1 0019C       CMPW    R0, #21844                               ;
                                            0C  12 001A1       BNEQ    14$                                      ;
                                            5E  DD 001A3 13$:   PUSHL   SP                                       ; 0895
                                       08   AE  9F 001A5       PUSHAB  Q_FILEBLOCK                              ;
                      FC07  CB              02  FB 001A8       CALLS   #2, ANSI_DIRECT                          ;
                                            07  11 001AD       BRB     15$                                      ;
                            50 000184C4     8F  D0 001AF 14$:   MOVL    #99524, R0                               ; 0896
                            52              50  D0 001B6 15$:   MOVL    R0, STATUS                               ; 0889
                                            10  12 001B9 16$:   BNEQ    17$                                      ; 0902
                                            53  DD 001BB       PUSHL   DEVPTR                                   ; 0905
                                            54  DD 001BD       PUSHL   DEVLEN                                   ;
                                       3E   AA  9F 001BF       PUSHAB  P.ABD                                    ;
                00000000G  9F               03  FB 001C2       CALLS   #3, @#DS$PRINTF                          ;
```

```
                                        06  11 001C9          BRB     18$                                     ;
                                50      52  D0 001CB 17$:     MOVL    STATUS, R0                              ; 0907
                                      0000G 30 001CE          BSBW    DSR$COMPLETION                          ;
                                18      08  A9  E8 001D1 18$: BLBS    CONTROLCFLAG, 19$                       ; 0909
                                        7E  7C 001D5          CLRQ    -(SP)                                   ; 0911
                        00000000G       9F  02 FB 001D7       CALLS   #2, @#DS$CNTRLC                         ;
                                        0C  52 F9 001DE       BLBC    STATUS, 19$                             ; 0913
                                        6E  DD 001E1          PUSHL   COUNT                                   ; 0915
                                    6C  AA  9F 001E3          PUSHAB  P.ABE                                   ;
                        00000000G       9F  02 FB 001E6       CALLS   #2, @#DS$PRINTF                         ;
                                            04 001ED 19$:     RET                                             ; 0917

; Routine Size:  494 bytes,    Routine Base:  CODE + 074D

;   0918  1
```

```
;    0919  1     %sbttl 'Master routine'
;    0920  1
;    0921  1     global routine dsv$directory : jsb_fake novalue =
;    0922  2         begin
;    0923  2         !
;    0924  2         ! Initialize context to prevent continuation of diagnostic or script.
;    0925  2         !
;    0926  2             script$flush ();                        ! Flush any active scripts
;    0927  2             ds_cleanup ();                          ! Force abort of diagnostic if any
;    0928  2             init_context ();                        ! Clean up stacks, etc.
;    0929  3             begin
;    0930  3
;    0931  3             builtin
;    0932  3                 sp;                                 ! 'Fake' return address to avoid trouble when BLISS does RSB
;    0933  3
;    0934  3             sp = .sp - 4;                           ! Do a PUSHAB L^BEGIN_BLISS
;    0935  3             .sp = begin_bliss                       !  (Set address for RSB)
;    0936  2             end;
;    0937  2             dsx$directory (ds$gl_clibase)
;    0938  1             end;                                    ! of dsv$directory


                        00000000G  EF   16 00000 DSV$DIRECTORY::
                                                 JSB     SCRIPT$FLUSH                    ; 0926
                        00000000G  EF   16 00006         JSB     DS_CLEANUP              ; 0927
                        00000000G  EF   16 0000C         JSB     INIT_CONTEXT            ; 0928
                     5E                 04   C2 00012    SUBL2   #4, SP                  ; 0934
                     6E 00000000G  EF   9E   00015       MOVAB   BEGIN_BLISS, (SP)       ; 0935
                     52 00000000G  EF   9E   0001C       MOVAB   DS$GL_CLIBASE, R2       ; 0937
                  FDEA  CF              00   FB 00023    CALLS   #0, DSX$DIRECTORY       ;
                                             05   00028    RSB                             ; 0938

; Routine Size:  41 bytes,    Routine Base:  CODE + 093B

;    0939  1
;    0940  1     end                                         ! End of module
;    0941  1
;    0942  0     eludom



;                                PSECT SUMMARY
;
;        Name                    Bytes                           Attributes
;
;   DATA                           456  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   WORK                            10  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   CODE                          2404  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                               Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total     Loaded   Percent     Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265              784         20         2         85        00:00.2
;  DRB1:[DS.WORK]DS.L32;159                653         15         2         42        00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7         18017        101         0        975        00:04.6




;                                       COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE DIRECTORY

; Size:          2404 code + 466 data bytes
; Run Time:         00:43.3
; Elapsed Time:     01:34.9
; Lines/CPU Min:     1305
; Lexemes/CPU-Min: 23491
; Memory Used:  233 pages
; Compilation Complete
```

Table of contents

(1) 144 Libraries and Symbol Definitions
(1) 171 Work Declarations
(1) 181 Data Declarations
(1) 227 Dispatch Routine
(1) 487 DSX$EndPass Routine
(1) 563 DS_Cleanup Routine
(1) 639 DSX$DoSummary and VRSummary Routines
(1) 698 VRShowStatus Routine

```
                        0000     1          .TITLE  DISPAT  *** DISPAT Diagnostic test sequence control
                        0000     2          .IDENT  /07-37/
                        0000     3          .NoShow Conditionals
                        0000     4
                        0000     5  ;
                        0000     6  ; Copyright (c) 1977, 1982
                        0000     7  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000     8  ;
                        0000     9  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    10  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    11  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    12  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    13  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    14  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    15  ; REMAIN IN DEC.
                        0000    16  ;
                        0000    17  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    18  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    19  ; CORPORATION.
                        0000    20  ;
                        0000    21  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    22  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    23
                        0000    24  ;++
                        0000    25  ; FACILITY:
                        0000    26  ;       VAX DIAGNOSTIC SUPERVISOR.
                        0000    27  ;
                        0000    28  ; ABSTRACT:
                        0000    29  ;
                        0000    30  ; ENVIRONMENT:
                        0000    31  ;
                        0000    32  ; AUTHOR:
                        0000    33  ;       KEN CHAPMAN     10-NOV-77       VERSION 01.
                        0000    34  ;
```

```
                                ;
                                ; MODIFIED BY:
                                ;
                                ;       TOM SOUTTER     14-NOV-77       VERSION 02.
                                ;       DSPR #24 & #25 -- CLEANUP SEQUENCING PROBLEMS.
                                ;       N.  HOWGATE     16-NOV-77       VERSION 03.
                                ;       DSPR #42 -- APT FUNCTION UPDATE
                                ;       TOM SOUTTER     08-DEC-77       VERSION 04.
                                ;       TURNED ON EXECUTION INFO. MESSAGES.
                                ;       DSPR #43 -- AUTOMATIC ''DOSUMMARY'' IN FINAL ''ENDPASS''.
                                ;       DSPR #64 -- CHECK LOAD FLAG FOR SUMMARY COMMAND.
                                ;       SET EXECUTE FLAG BEFORE CHECKING EXIT TEST STATUS. (KWC)
                                ;       NICK HOWGATE    30-DEC-77       VERSION 05.
                                ;       DSX$ENDPASS - CHANGE REFERENCE TO DSA$GL_PASSES
                                ;       Roger Riggs     12-Jun-78       VERSION 08
                                ;       Standardizing primary messages (first pass done/end run)
                                ;       Check flag DS$V_DONFLG in DS$GL_FLAGS so cleanup is done
                                ;       only once.
                                ;       Fix to use .ENTRY and .VECTOR in proceedures
                                ;       Roger Riggs     12-Jun-1978
                                ;       Removed deallocate from cleanup code.
                                ;       Cancel ^O before calling summary  and end pass message
                                ;       Dave Butenhof   15-feb-80
                                ;       Add hooks for 'ERROR SEARCH' function
                                ;       Roger Riggs,    12-Mar-1980
                                ;       Added call to INITSCB and reset memory management after cleanup
                                ;       also call QIO$CLEANUP only if program is SEP_FUNCTIONAL
                                ;
                                ;       Dave Butenhof   30-jun-1980, version 5.5
                                ;       Correct word displaced mode to longword displaced
                                ;       Support DS RMS by calling cleanup routine to avoid
                                ;       cluttering pool space in case of errors
                                ;
                                ;       Dave Butenhof, 03-oct-1980, version 6.1
                                ;       Change some more word-relative references to longword relative
                                ;       Clear DS$V_CTRLC bit before doing $DS_CNTRLC call; this is to
                                ;       prevent probably undesired control C at next call to KB_CHECK,
                                ;       if control C's were disabled by previous program.
                                ;       Dave Butenhof, 29-dec-1980, version 6.2
                                ;       More word relative references to longword.  This is getting
                                ;       quite un-fun.
                                ;       Dave Butenhof, 17-feb-1981, version 6.3
                                ;       More of same.  ALL DSA$?_? references this time.
                                ;       - dave butenhof, 06-Oct-1981, version 6.5
                                ;       Add SHOW STATUS command
                                ;       - Dave Butenhof, 02-Nov-1981, version 6.-
                                ;       Fix truncation errors
```

```
0000     82 ;        22      - Jack Stansbury, 03-Nov-1981, Version 6.-
0000     83 ;                Added calls to QA$Main for the QA enhancements. Also fixed
0000     84 ;                some truncation errors. Also changed PSECTs a bit. Added
0000     85 ;                .LIBRARY statements for $DS, $DIAG, and LIB.
0000     86 ;
0000     87 ;        23      - Dave Butenhof, 16-Nov-1981, version 6.5
0000     88 ;                Modified the text (and corrected a bug) in the SHOW STATUS
0000     89 ;                command.
0000     90 ;
0000     91 ;        24      - Dave Butenhof, 19-Nov-1981, version 6.5
0000     92 ;                Modify to use typecodes on all output messages.
0000     93 ;
0000     94 ;        25      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000     95 ;                Commented out several of the calls to QA_MAIN because it
C000     96 ;                appears as though they will not be needed for the 6.5 DS
0000     97 ;                version. Made several other small code changes to the DISPATCH
0000     98 ;                routine for QA.
0000     99 ;
0000    100 ;        26      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000    101 ;                Changed DSX$ENDPASS routine slightly to accomadate QA.
0000    102 ;
0000    103 ;        27      - Jack Stansbury, 16-Dec-1981, Version 6.6
0000    104 ;                Added the DS_QADEFS macro definition.
0000    105 ;
0000    106 ;        28      - Dave Butenhof, 29-Dec-1981, Version 6.6
0000    107 ;                Print out all error counts on Summary command.
0000    108 ;
0000    109 ;        29      - Jack Stansbury, 6-Jan-1982, Version 6.6
0000    110 ;                Changed EndPass routine slightly to accomodate the
0000    111 ;                new way of implementing QA (with the QA Start Loop).
0000    112 ;
0000    113 ;        30      - Dave Butenhof, 12-Jan-1982, Version 6.6
0000    114 ;                Fix EndPass message to avoid losing end of message on
0000    115 ;                narrow screen.
0000    116 ;
0000    117 ;        31      - Jack Stansbury, 15-Jan-1982, Version 6.6
0000    118 ;                Made changes to the Dispatch routine for the Loop on Subtest
0000    119 ;                QA check routine.
0000    120 ;
0000    121 ;        32      - Jack Stansbury, 9-Mar-1982, Version 6.6
0000    122 ;                Inserted some new DS macros that will help these routines
0000    123 ;                be more understandable.
0000    124 ;
0000    125 ;        33      Marion Baggett, 9-Apr-1982, Version 6.7
0000    126 ;                Added NoShow Conditionals after the .IDENT
0000    127 ;
0000    128 ;        [34] Dave Butenhof, 20-Apr-1982, Version 6.7
0000    129 ;                Add kluge code in DS_CLEANUP; if standalone on 730,
0000    130 ;                reset IDC to prevent extra interrupts.
0000    131 ;
0000    132 ;        35      Jack Stansbury, 13-September-1982, Version 6.9
0000    133 ;                Fixed some truncation errors. I didn't mark those that I fixed.
0000    134 ;
0000    135 ;        36      Jack Stansbury, 15-September-1982, Version 6.9
0000    136 ;                Changed the VRSummary and VRShowStatus routines to call the
0000    137 ;                DSR$CheckLoad routine defined in ShowMem.
0000    138 ;
```

```
0000   139 ;       37      Jack Stansbury, 8-Mar-1983, Version 6.11
0000   140 ;               Changed the VRShowStatus routine to look for the user's PC
0000   141 ;               on the stack (in the CLI args).
0000   142 ;--
```

```
                        0000   144          .SBTTL  Libraries and Symbol Definitions
                        0000   145 ;
                        0000   146 ; INCLUDE FILES:
                        0000   147 ;
                        0000   148
                        0000   149          .LIBRARY        /SYS$LIBRARY:LIB/       ;                       [22]
                        0000   150          .LIBRARY        /$DS/                   ;                       [22]
                        0000   151          .LIBRARY        /$DIAG/                 ;                       [22]
                        0000   152 ;
                        0000   153 ; MACROS:
                        0000   154 ;
                        0000   155
                        0000   156 ;
                        0000   157 ; EQUATED SYMBOLS:
                        0000   158 ;
                        0000   159          $PrDef                          ; Define PR codes               [34]
                        0000   160          $DS_ENVDEF
                        0000   161          $DS_DSADEF
                        0000   162          $DS_HDRDEF
                        0000   163          APIDEF
                        0000   164          CLIDEF                  ; Cli$L_ definitions                    [31]
                        0000   165          DSFDEF
                        0000   166          DSPDEF
                        0000   167          DSQA                    ; DSQA$K constant definitions          [22]
                        0000   168          DS_QADEFS               ; Other QA$K constant definitions      [25]
                        0000   169          $DS_TYPEDEF             ; Define type codes                    [24]
```

```
                    0000   171          .SUBTITLE       Work Declarations
                00000000   172          .PSECT  Work, NoExe, NoShr, Wrt, Long          ;                    [22]
                    0000   173
                    0000   174 ;
                    0000   175 ; OWN STORAGE:
                    0000   176 ;
                    0000   177
00000004            0000   178 A_TESTPTR:       .BLKL   1       ; POINTER TO CURRENT TEST CODE.
00000008            0004   179 A_DATAPTR:       .BLKL   1       ; POINTER TO CURRENT TEST DATA.
```

```
                                    0008   181          .SUBTITLE       Data Declarations
                                00000000   182          .PSECT  Data, NoExe, Shr, NoWrt, Long          ;                    [22]
                                    0000   183
                                    0000   184          MODNAM  DISPATCH
                                    0009   185
                                    0009   186 T_NOEXE:
73 65 74 20 6F 4E 20 3F 3F 2F 21 00'0009   187          .ASCIC  \!/?? No tests in this section.!/\      ;                    [24]
73 20 73 69 68 74 20 6E 69 20 73 74 0015
          2F 21 2E 6E 6F 69 74 63 65 0021
                                20    0009
                                    002A   188
                                    002A   189 FMTPASONE:
20 74 73 72 69 46 20 2E 2E 2F 21 00'002A   190          .ASCIC  "!/.. First pass done, !UW error!%S detected, time is !%D!/" ;  [30]
21 20 2C 65 6E 6F 64 20 73 73 61 70 0036
20 53 25 21 72 6F 72 72 65 20 57 55 0042
69 74 20 2C 64 65 74 63 65 74 65 64 004E
   2F 21 44 25 21 20 73 69 20 65 6D 005A
                                3A    002A
                                    0065   191
                                    0065   192 FMTLSTPAS:
66 6F 20 64 6E 45 20 2E 2E 2F 21 00'0065   193          .ASCIC  "!/.. End of run, !UW error!%S detected, pass count is !UW.!/"-;[30]
72 65 20 57 55 21 20 2C 6E 75 72 20 0071
63 65 74 65 64 20 53 25 21 72 6F 72 007D
6F 63 20 73 73 61 70 20 2C 64 65 74 0089
21 2C 57 55 21 20 73 69 20 74 6E 75 0095
20 73 69 20 65 6D 69 74 20 20 20 2F 00A1
                  2F 21 44 25 21 00AD
                                4C    0065
                                    0082   194                  "   time is !%D!/"                 ;                    [30]
```

```
                                                00B2   196 FMTSEARCH:
20 72 6F 72 72 45 20 2E 2E 2F 21 00'            00B2   197         .ASCIC  "!/.. Error search: new test limit is !UL, time is !%D!/" ;     [30]
20 77 65 6E 20 3A 68 63 72 61 65 73             00BE
69 20 74 69 6D 69 6C 20 74 73 65 74             00CA
20 65 6D 69 74 20 2C 4C 55 21 20 73             00D6
                  2F 21 44 25 21 20 73 69       00E2
                                    37          00B2
                                                00EA   198
                                                00EA   199 FMTTRACE:
3A 42 55 21 20 74 73 65 54 2F 21 00'            00EA   200         .ASCIC  "!/Test !UB: !AC"
                              43 41 21 20       00F6
                                    0F          00EA
                                                00FA   201
                                                00FA   202 FMT_SUMMARY:                                 ;                       [28]
6F 20 79 72 61 6D 6D 75 53 2F 21 00'            C0FA   203         .ascic  "!/Summary of !AC, Rev !UL.!UL:!/" -     ;                       [28]
21 20 76 65 52 20 2C 43 41 21 20 66             0106
                  2F 21 3A 4C 55 21 2E 4C 55    0112
6D 61 72 67 6F 72 70 20 4C 5A 21 20             011B   204                 " !ZL program detected error!%S " -     ;                       [28]
72 65 20 64 65 74 63 65 74 65 64 20             0127
                  20 53 25 21 72 6F 72          0133
21 20 2C 64 72 61 48 20 4C 5A 21 28             013A   205                 "(!ZL Hard, !ZL Soft, !ZL" -            ;                       [28]
4C 5A 21 20 2C 74 66 6F 53 20 4C 5A             0146
4C 5A 21 20 2C 6D 65 74 73 79 53 20             0152   206                 " System, !ZL Device).!/" -             ;                       [28]
   2F 21 2E 29 65 63 69 76 65 44 20             015E
69 76 72 65 70 75 53 20 4C 5A 21 20             0169   207                 " !ZL Supervisor detected error!%S.!/!/"        ;               [28]
64 65 74 63 65 74 65 64 20 72 6F 73             0175
2F 21 2E 53 25 21 72 6F 72 72 65 20             0181
                                 2F 21          018D
                                    94          00FA
```

```
                                        018F    209 Fmt_ShowStatus_No_PC:                                       ;                               [37]
21 20 66 6F 20 73 75 74 61 74 53 00'    018F    210         .Ascic  "Status of !AC, Rev !UL.!UL!/" -            ;                               [37]
2E 4C 55 21 20 76 65 52 20 2C 43 41     019B
                        2F 21 4C 55 21  01A7
69 74 63 65 53 5B 5F 21 44 25 21 20     01AC    211                 " !%D!_[Section !AC]!_[!ZL error!%S]!/" -       ;                       [37]
5A 21 5B 5F 21 5D 43 41 21 20 6E 6F     01B8
21 5D 53 25 21 72 6F 72 72 65 20 4C     01C4
                              2F        01D0
21 5D 4C 5A 21 20 74 73 65 54 5B 20     01D1    212                 " [Test !ZL]!_[Subtest !ZL]!/" -        ;       [37]
5A 21 20 74 73 65 74 62 75 53 5B 5F     01DD
                        2F 21 5D 4C     01E9
21 5D 4C 5A 21 20 73 73 61 50 5B 20     01ED    213                 " [Pass !ZL]!/"                         ;       [37]
                              2F        01F9
                              6A        018F
                                        C1FA    214
                                        01FA    215 FMT_SHOWSTATUS:
21 20 66 6F 20 73 75 74 61 74 53 00'    01FA    216         .ascic  "Status of !AC, Rev !UL.!UL!/" -            ;                               [23]
2E 4C 55 21 20 76 65 52 20 2C 43 41     0206
                        2F 21 4C 55 21  0212
69 74 63 65 53 5B 5F 21 44 25 21 20     0217    217                 " !%D!_[Section !AC]!_[!ZL error!%S]!/" -       ;                       [23]
5A 21 5B 5F 21 5D 43 41 21 20 6E 6F     0223
21 5D 53 25 21 72 6F 72 72 65 20 4C     022F
                              2F        023B
21 5D 4C 5A 21 20 74 73 65 54 5B 20     023C    218                 " [Test !ZL]!_[Subtest !ZL]!/" -        ;       [23]
5A 21 20 74 73 65 74 62 75 53 5B 5F     0248
                        2F 21 5D 4C     0254
21 5D 4C 5A 21 20 73 73 61 50 5B 20     0258    219                 " [Pass !ZL]!_[User PC !XL]!/"  ;               [23]
58 21 20 43 50 20 72 65 73 55 5B 5F     0264
                        2F 21 5D 4C     0270
                              79        01FA
                                        0274    220
                                        0274    221 FMT_NODIAG:
6F 67 61 69 64 20 6F 4E 20 3F 3F 00'    0274    222         .ascic  '?? No diagnostic running.!/'   ;               [23]
6E 69 6E 6E 75 72 20 63 69 74 73 6F     0280
                        2F 21 2E 67     028C
                              1B        0274
                                        0290    223
                                        0290    224 T_TSTABO:
   5D 64 65 74 72 6F 62 61 5B 20 00'    0290    225         .ASCIC  " [aborted]"
                              0A        0290
```

```
                     0298   227          .SBTTL  Dispatch Routine
                 00000000   228          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                [22]
                     0000   229 ;++
                     0000   230 ; FUNCTIONAL DESCRIPTION:
                     0000   231 ;
                     0000   232 ;       This routine is called to initiate the test program.
                     0000   233 ;       It takes case of calling initialization sections, individual
                     0000   234 ;       tests and finally cleanup code.
                     0000   235 ;
                     0000   236 ; CALLING SEQUENCE:
                     0000   237 ;
                     0000   238 ;       BSBW    DISPATCH
                     0000   239 ;
                     0000   240 ; INPUT PARAMETERS:
                     C000   241 ;
                     0000   242 ;       DS$GL_FSTTEST   = FIRST TEST NUMBER
                     0000   243 ;       DS$GL_LSTTEST   = NUMBER OF LAST TEST TO EXECUTE
                     0000   244 ;
                     0000   245 ; IMPLICIT INPUTS:      NONE
                     0000   246 ;
                     0000   247 ; OUTPUT PARAMETERS:    NONE
                     0000   248 ;
                     0000   249 ; IMPLICIT OUTPUTS:     NONE
                     0000   250 ;
                     0000   251 ; COMPLETION CODES:     NONE
                     0000   252 ;
                     0000   253 ; SIDE EFFECTS:         NONE
                     0000   254 ;--
```

```
                        0000   256 DISPATCH::
0C000000'EF    D4       0000   257         CLRL    L^DS$GL_ERRCNT          ; START WITH NO ERRORS
00000000'EF    D4       0006   258         Clrl    L^Ds$GL_HardErr_Count   ; Clear out error counters              [28]
00000000'EF    D4       000C   259         Clrl    L^Ds$GL_SoftErr_Count   ;                                       [28]
00000000'EF    D4       0012   260         Clrl    L^Ds$GL_SysErr_Count    ;                                       [28]
00000000'EF    D4       0018   261         Clrl    L^Ds$GL_DevErr_Count    ;                                       [28]
00000000'EF    D4       001E   262         Clrl    L^Ds$GL_ErrSup_Count    ;                                       [28]
0000FE54'EF    D4       0024   263         CLRL    L^DSA$GL_PASSNO         ; START ON FIRST PASS
                        002A   264         Set_Pass0                       ; Indicate first pass ever              [32]
                        0032   265
                        0032   266 FIRSTTEST:
                        0032   267
                        0032   268         ;+
                        0032   269         ; Perform the initialization code for the diagnostic.
                        0032   270         ; Test 0, Subtest 1 => initialization code
                        0032   271         ; Test 0, Subtest 2 => cleanup code
                        0032   272         ;-
                        0032   273
0000FE50'EF    D4       0032   274         CLRL    L^DSA$GL_TESTNO         ; TEST 0,
0000FE4C'EF    01       0038   275         MOVL    #1, L^DSA$GL_SUBTNO     ; SUBTEST 1 INDICATES INIT.
                        003F   276
                        003F   277         QA_MAIN Dispat_Before_Init      ; Call QA$Main                          [22]
                        0048   278
0000023C'FF    C0       0048   279         CALLS   #0, @L$A_ICP            ; Execute initialization code.
0000FE50'EF    00000000'EF D0  004F   280         MOVL    L^DS$GL_FSTTEST, -      ; LOAD NUMBER OF FIRST TEST
                        005A   281                   L^DSA$GL_TESTNO
                        005A   282
                        005A   283         QA_MAIN Dispat_After_Init       ; Call QA$Main                          [22]
                        0063   284
0000FE4C'EF    D4       0063   285         CLRL    L^DSA$GL_SUBTNO         ; RESET SUBTEST NUMBER.
0000FE00'EF    1D       0069   286         BBCC    #DSA$V_PASS0, -         ; Clear first pass flag.
               07       0070   287                   L^DSA$GL_FLAGS, CALLTEST
0000FE54'EF    01       0071   288         MOVL    #1, L^DSA$GL_PASSNO     ; START ON FIRST PASS
```

```
                                0078   290 CALLTEST:
                                0078   291
                                0078   292         ;+
                                0078   293         ; Set up everything to call a test in the diagnostic.
                                0078   294         ;-
                                0078   295
                                0078   296         QA_MAIN Dispat_CallTest          ; Call QA$Main                          [22]
                                0081   297         Clear_ErrFlg                     ; Clear the error flag                  [32]
                                0089   298
                                0089   299         ;+
                                0089   300         ; Compute the starting address of the next test to execute.
                                0089   301         ; Use the DSA$GL_TESTNO to compute the address. If no more tests,
                                0089   302         ; go to 130$ label.
                                0089   303         ;-
                                C089   304
      5C   0000FE50'EF    01   C3  0089   305         SUBL3   #1, L^DSA$GL_TESTNO, AP          ; GET INDEX INTO DISPATCH TBL
                   5C    18   C4  0091   306         MULL2   #DSP$K_SIZE, AP
         5C   00000218'FF4C   9E  0094   307         MOVAB   @L$A_DTP[AP], AP                 ; GET ADDRESS OF DISP TBL ENTRY.
   00000000'EF    0C AC    D0  009C   308         MOVL    DSP$A_TEST(AP), -                ; GET TEST CODE POINTER.
                                00A4   309                   L^A_TESTPTR
                      03    12  00A4   310         BNEQ    20$                              ; If not equal to zero, branch
                    01C7    31  00A6   311         BRW     130$                             ; Was equal to zero
                                00A9   312
   00 BC   0000FE50'EF    91  00A9   313 20$:     CMPB    L^DSA$GL_TESTNO, -               ; CHECK TEST NUMBER.
                                00B1   314                   @DSP$A_BASE(AP)
                      14    13  00B1   315         BEQL    30$                              ; If they are equal, branch
                                00B3   316         ERRSUP_S                                 ; Otherwise, error in DS
                    01D8    31  00C4   317         BRW     DISPATCH_X
                                00C7   318
 05 10 AC   0000FE10'EF    E0  00C7   319 30$:     BBS     L^DSA$GL_SECTNO, -               ; Check for right section
                                00D0   320                   DSP$Q_SECTION(AP), 35$
                    00F2    31  00D0   321         BRW     115$                             ; Branch if bit clear
```

```
                                00D3   323          ;+
                                00D3   324          ; If TRACE flag is set, print the trace message.
                                00D3   325          ;-
                                00D3   326
                                00D3   327 35$:     Br_If_Not_Trace 40$                         ; If not tracing, branch          [32]
                                00DB   328          $PRINT  #ds$k_type_program_info,-           ; Print info message              [24]
                                00DB   329                  #ds$k_printf, -                     ; .. use PRINTF                   [24]
                                00DB   330                  L^FMTTRACE, -                       ; .. Trace message                [24]
                                00DB   331                  @DSP$A_BASE(AP), -                  ; .. test number                  [24]
                                00DB   332                  DSP$A_MSG(AP)                       ; .. and test name                [24]
                                00F4   333
      00000004'EF    08 AC    D0  00F4   334 40$:     MOVL    DSP$A_DATA(AP), -                   ; GET DATA TABLE ENTRY.
                                00FC   335                  L^A_DATAPTR
                                00FC   336
            00000000'EF    D4  00FC   337 50$:     CLRL    L^DS$GA_CHKLPPC                     ; CLEAR THE CHECK_LOOP P.C.
            0000FE4C'EF    D4  0102   338          CLRL    L^DSA$GL_SUBTNO                     ; START WITH SUBTEST ZERO
                                0108   339          Clear_Subt                                  ; Clear the subtest flag          [32]
                                0110   340
                                0110   341          ;+
                                0110   342          ; Call the test that is numbered DSA$GL_TESTNO.
                                0110   343          ;-
                                0110   344
   00000000'FF    00000004'FF  FA  0110   345          CALLG   @L^A_DATAPTR, -                     ; CALL THE TEST.
                                0118   346                  @L^A_TESTPTR
                                0118   347
                                0118   348          ;+
                                0118   349          ; The test will come back to here assuming there were
                                0118   350          ; no strange escapes from the test.
                                0118   351          ;-
                                0118   352
                                0118   353          Set_ExeTst                                  ; Set execute flag                [32]
                                0123   354          Br_If_Not_ErrFlg        100$               ; Branch if no errors             [32]
                                0128   355          Br_If_Not_Search        100$               ; Branch if not searching         [32]
```

```
                                0133   357          ;+
                                0133   358          ; This section is for performing SEARCHing using the search flag.
                                0133   359          ;-
                                0133   360
                                0133   361          Clear_ErrFlg                                    ; Clear the error flag          [32]
00000000'EF   0000FE50'EF   D0  013B   362          MOVL    L^DSA$GL_TESTNO,L^DS$GL_LSTTEST         ; Copy test number to limit
FFFFFFFF 8F   00000000'EF   F1  0146   363          ACBL    L^DS$GL_FSTTEST,#-1, -                  ; Decrement last test, and be
              00000000'EF       0151
                          0012  0156   364                    L^DS$GL_LSTTEST,80$                   ;   sure it's in range--else bad
0000FE08'EF   0000FE54'EF   D0  0158   365          MOVL    L^DSA$GL_PASSNO,L^DSA$GL_PASSES         ; Pretend we just did last pass
         000002A0'EF   00   FB  0163   366          CALLS   #0,L^DSX$ENDPASS                        ; Now 'fake' an EOP to stop testing
                                016A   367                                                          ; DSX$ENDPASS will not return here!
                                016A   368
                                016A   369 80$:      $PRINT  #ds$k_type_program_info, -              ; Print search message          [24]
                                016A   370                    #ds$k_printf, -                       ; .. use PRINTF                 [24]
                                016A   371                    L^FMTSEARCH, -                        ; .. SEARCH message             [24]
                                016A   372                    L^DS$GL_LSTTEST, -                    ; .. test number                [24]
                                016A   373                    #0                                    ; .. and current time           [24]
```

```
                        0185   375          ;+
                        0185   376          ; If the test did an ABORT, R0 will be clear. If it is clear and the
                        0185   377          ; trace flag is set, print a message saying that the test was aborted
                        0185   378          ; and go on with the next test in the sequence.
                        0185   379          ;-
                        0185   380
                 1D 50  E8      0185   381 100$:   BLBS    R0, 110$                        ; If success, branch
                        0188   382          Br_If_Not_Trace 115$                          ; If not tracing, branch around [32]
                        0190   383          $print  #ds$k_type_abort_test,-                ; Print abort test message      [24]
                        0190   384                  #ds$k_printi, -                       ; .. use PRINTI (was TYPMSG)    [24]
                        0190   385                  L^T_TSTABO                            ; .. ABORT text                 [24]
                     20 11     01A3   386          BRB     115$
                        01A5   387
                        01A5   388          ;+
                        01A5   389          ; Check to see if the same test should be called again with
                        01A5   390          ; another data argument list. If so, compute address of new
                        01A5   391          ; data and go back to call the same test again.
                        01A5   392          ;-
                        01A5   393
           50   00000004'EF  D0      01A5   394 110$:   MOVL    L^A_DATAPTR, R0                 ; GET POINTER TO ARG LIST
                     51 60    D0      01AC   395         MOVL    (R0),R1                         ; GET COUNT OF ARGS IN LIST
                     14 13    DE      01AF   396         BEQL    115$                            ; END OF DATA LIST, DO NEXT TEST
00000004'EF   04 A041  DE      01B1   397         MOVAL   4(R0)[R1], L^A_DATAPTR          ; POINT PAST ARG LIST
                00000004'FF D5 01BA   398         TSTL    @L^A_DATAPTR                    ; END OF LIST OF ARG LISTS?
                     03 13    01C0   399         BEQL    115$                            ; Yes--next test
                   FF37 31    01C2   400         BRW     50$                             ; Repeat test with next arg list
```

```
                                 01C5   402          ;+                                                                  [25]
                                 01C5   403          ; Check to see if QA is running AND if several cases of check       [25]
                                 01C5   404          ; routines are currently executing. If both are true, perform special [25]
                                 01C5   405          ; code to accomplish the checks.                                    [25]
                                 01C5   406          ;-
                                 01C5   407
                                 01C5   408 115$:    Br_If_QA        117$                ; If running QA, branch         [32]
                          0092   31    01CD   409          BRW     125$                ; QA is not set, branch around. [31]
                                 01D0   410
          00000000'EF     03     E1    01D0   411 117$:    BBC     S^#QA$K_Loop_On_Subtest, -  ; If the Loop on Subtest check [31]
                                 5B    01D7   412            L^QA$AOB_Check_State, 122$  ; is not running, branch.       [31]
                                 01D8   413
                                 01D8   414          ;+
                                 01D8   415          ; This is for the Loop on Subtest check only.                       [31]
                                 01D8   416          ; We have finished running one test and all its subtests. Reset the [31]
                                 01D8   417          ; subtest number, and add one to both the first and last test numbers. [31]
                                 01D8   418          ; Make these changes in both the CLI data vector and the DS (DSA)   [31]
                                 01D8   419          ; data areas.                                                       [31]
                                 01D8   420          ;-
                                 01D8   421
          51    0000FE50'EF      D0    01D8   422          MOVL    L^DSA$GL_TestNo, R1         ; Get the current test number. [31]
          50    0000FE08'EF      D0    01DF   423          MOVL    L^DSA$GL_PASSES, R0         ; Save the Passes count.       [31]
          0000FE08'EF     01     D0    01E6   424          MOVL    #1, L^DSA$GL_PASSES         ; Set to 1 in case we take the [31]
                                 01ED   425                                                  ; branch to 130$.              [31]
    0002  51    01    00000000'EF      3D    01ED   426          ACBW    L^QA$W_Save_Last, #1, R1, 121$ ; Branch if (R1 + 1) LEQ the [31]
                                 01F7   427                                                  ; saved last test number.      [31]
                          77     11    01F7   428          BRB     130$                ; Otherwise, done all the tests.[31]
                                 01F9   429
    0000FE08'EF     50     D0    01F9   430 121$:    MOVL    R0, L^DSA$GL_PASSES         ; Restore Passes count.        [31]
    50    00000000'EF      DE    0200   431          MOVAL   L^DS$GL_CLIBASE, R0         ; Address the CLI data vector. [31]
    00000028 E0    01     D0    0207   432          MOVL    #1, L^CLI$L_SUBT (R0)       ; Set the Subtest number to 1. [31]
    00000000'EF    01     D0    020E   433          MOVL    #1, L^DS$GL_SUBTEST         ; Ditto.                       [31]
                                 0215   434
    00000020 E0    51     D0    0215   435          MOVL    R1, L^CLI$L_TEST (R0)       ; Increment First test.        [31]
    00000000'EF    51     D0    021C   436          MOVL    R1, L^DS$GL_FSTTEST         ; Ditto.                       [31]
    00000024 E0    51     D0    0223   437          MOVL    R1, L^CLI$L_LAST (R0)       ; Increment Last test.         [31]
    00000000'EF    51     D0    022A   438          MOVL    R1, L^DS$GL_LSTTEST         ; Ditto.                       [31]
                          2F     11    0231   439          BRB     125$                ; Go execute another test.     [31]
```

B 1 CONVERT BINARY TIME TO NUMERIC
C 1 CONVERT BINARY TIME TO NUMERIC
D 1 CONVERT BINARY TIME TO NUMERIC
E 1 CONVERT BINARY TIME TO NUMERIC
F 1 Symbol table
G 1 Cross reference
H 1 Cross reference
I 1 Cross reference
J 1 - Converts ASCII to RAD50
K 1 - Converts ASCII to RAD50
L 1 DECLARATIONS
M 1 CVTFILNAM - CONVERT FILE NAME
N 1 CVTFILNAM - CONVERT FILE NAME
B 2 CVTFILNAM - CONVERT FILE NAME
C 2 STORER50BYTE - CONVERT AND STO
D 2 PACKRAD50 - PACK 3 BYTES OF RA
E 2 ASCIITORAD50 - CONVERT ASCII C
F 2 Symbol table
G 2 Cross reference
H 2 Cross reference
I 2 *** DASSGN deassign QIO channe
J 2 *** DASSGN deassign QIO channe
K 2 *** DASSGN deassign QIO channe
L 2 DEASSIGN I/O CHANNEL
M 2 DEASSIGN I/O CHANNEL
N 2 Symbol table
B 3 Symbol table
C 3 Psect synopsis
D 3 Cross reference
E 3 Cross reference
F 3 Cross reference
G 3 - CONSOLE TU58 BOOT DRIVER
H 3 - CONSOLE TU58 BOOT DRIVER
I 3 - CONSOLE TU58 BOOT DRIVER
J 3 DECLARATIONS
K 3 DECLARATIONS
L 3 DD_SELECT - Select correct CPU
M 3 Console TU58 Driver
N 3 Console TU58 Driver
B 4 Console TU58 Driver
C 4 Console TU58 Driver
D 4 Console TU58 Driver
E 4 Character transmission
F 4 Character transmission
G 4 Character transmission
H 4 Character transmission
I 4 Symbol table
J 4 Symbol table
K 4 Cross reference
L 4 Cross reference
M 4 Cross reference
N 4 Cross reference
B 5 Cross reference
C 5 EXAMINE, DEPOSIT, AND BREAKPOI
D 5 EXAMINE, DEPOSIT, AND BREAKPOI
E 5 EXAMINE, DEPOSIT, AND BREAKPOI
F 5 DECLARATIONS
G 5 DECLARATIONS
H 5 DECLARATIONS
I 5 DECLARATIONS
J 5 DECLARATIONS
K 5 DECLARATIONS
L 5 DECLARATIONS
M 5 SET BASE ADDRESS SUBROUTINE
N 5 SHOW BASE ADDRESS SUBROUTINE
B 6 SET DEFAULTS SUBROUTINE
C 6 SET DEFAULTS SUBROUTINE
D 6 SHOW DEFAULTS SUBROUTINE
E 6 SHOW DEFAULTS SUBROUTINE
F 6 EXAMINE MEMORY AND REGISTERS S
G 6 EXAMINE MEMORY AND REGISTERS S
H 6 EXAMINE MEMORY AND REGISTERS S
I 6 EXAMINE MEMORY AND REGISTERS S
J 6 VRTYPEXAMINE Type out location
K 6 VRTYPEXAMINE Type out location
L 6 VRTYPEXAMINE Type out location
M 6 VRTYPEXAMINE Type out location
N 6 DEPOSIT MEMORY AND REGISTERS S
B 7 DEPOSIT MEMORY AND REGISTERS S
C 7 DEPOSIT MEMORY AND REGISTERS S
D 7 DEPOSIT MEMORY AND REGISTERS S
E 7 SET BREAKPOINT SUBROUTINE
F 7 SET BREAKPOINT SUBROUTINE
G 7 CLEAR BREAKPOINT SUBROUTINE
H 7 CLEAR BREAKPOINT SUBROUTINE
I 7 DSP$REMOVEBREAK Remove breakpo
J 7 SHOW BREAKPOINTS SUBROUTINE
K 7 FIND_BPT FIND BPT IN BPT TABLE
L 7 DEBUGGER CONDITION HANDLER
M 7 BREAKPOINT CONDITION SUBROUTIN
N 7 BREAKPOINT CONDITION SUBROUTIN
B 8 BREAKPOINT CONDITION SUBROUTIN
C 8 TBIT CONDITION SUBROUTINE
D 8 TBIT CONDITION SUBROUTINE
E 8 TEMPORARY BREAKPOINT REMOVAL S
F 8 BREAKPOINT REPLACEMEMT SUBROUT
G 8 FETCH_STORE COPY BYTE ROUTINE
H 8 FETCH_STORE COPY BYTE ROUTINE
I 8 FETCH_STORE COPY BYTE ROUTINE
J 8 FETCH_STORE COPY BYTE ROUTINE
K 8 FETCH_STORE COPY BYTE ROUTINE
L 8 FETCH_STORE COPY BYTE ROUTINE
M 8 Handle EXAMINE/DEPOSIT of IPRs
N 8 FETCH_STOR_PREG Move data to/f
B 9 DSV$DEBUG - Process DEBUG Comm
C 9 DSV$DEBUG - Process DEBUG Comm
D 9 Symbol table
E 9 Symbol table
F 9 Symbol table
G 9 Symbol table
H 9 Cross reference
I 9 Cross reference
J 9 Cross reference
K 9 Cross reference
L 9 Cross reference
M 9 Cross reference
N 9 Cross reference
B 10 Cross reference
C 10 Cross reference
D 10 Cross reference
E 10 *** DEVALC device allocation r
F 10 *** DEVALC device allocation r
G 10 *** DEVALC device allocation r
H 10 ALLOCATE DEVICE
I 10 ALLOCATE DEVICE
J 10 ALLOCATE DEVICE
K 10 DEALLOCATE DEVICE
L 10 DEALLOCATE DEVICE
M 10 LOCK I/O DATA BASE AND SEARCH
N 10 DECREMENT REFERENCE COUNT, CLE
B 11 DSX$MOUNT - Mount a device
C 11 DSX$DISMOUNT - Dismount a devi
D 11 Symbol table
E 11 Symbol table
F 11 Symbol table
G 11 Cross reference
H 11 Cross reference
I 11 Cross reference
J 11 Cross reference
K 11 *** DEVICE Handle PTABLEs
L 11 *** DEVICE Handle PTABLEs
M 11 *** DEVICE Handle PTABLEs
N 11 DS$GPHARD Retreive address of
B 12 DSP$GEN_PTABLES Setup UUT's to
C 12 DSP$GEN_PTABLES Setup UUT's to
D 12 Symbol table
E 12 Psect synopsis
F 12 Cross reference
G 12 Cross reference
H 12 Cross reference
I 12 Cross reference
J 12 *** DIRECTORY Handle DIRECTORY
K 12 *** DIRECTORY Handle DIRECTORY
L 12 *** DIRECTORY Handle DIRECTORY
M 12 *** DIRECTORY Handle DIRECTORY
N 12 Put filename to buffer, if mat
B 13 Put filename to buffer, if mat
C 13 Put filename to buffer, if mat
D 13 RAD50
E 13 RAD50
F 13 format_ods1
G 13 format_ods1
H 13 format_ods2
I 13 format_ods2
J 13 format_ods2
K 13 RT-11 directory
L 13 RT-11 directory
M 13 RT-11 directory
N 13 RT-11 directory
B 14 ANSI directory
C 14 ANSI directory
D 14 ANSI directory
E 14 ANSI directory
F 14 ANSI directory
G 14 ods directory
H 14 ods directory
I 14 ods directory
J 14 ods directory
K 14 ods directory
L 14 ods directory

```
M 14  Main code
N 14  Main code
B 15  Main code
C 15  Main code
D 15  Main code
E 15  Main code
F 15  Main code
G 15  Master routine
H 15  Master routine
I 15  *** DISPAT Diagnostic test seq
J 15  *** DISPAT Diagnostic test seq
K 15  *** DISPAT Diagnostic test seq
L 15  *** DISPAT Diagnostic test seq
M 15  *** DISPAT Diagnostic test seq
N 15  Libraries and Symbol Definitio
B 16  Work Declarations
C 16  Data Declarations
D 16  Data Declarations
E 16  Data Declarations
F 16  Dispatch Routine
G 16  Dispatch Routine
H 16  Dispatch Routine
I 16  Dispatch Routine
J 16  Dispatch Routine
K 16  Dispatch Routine
L 16  Dispatch Routine
```

```
                00000000'EF   02   E1  0233   441 122$:   BBC     S^#QA$K_Loop_On_Test, -          ; If the Loop on Test check is  [25]
                              0B       023A   442             L^QA$AOB_Check_State, 123$       ; not running, branch           [25]
                00000000'EF   04   E0  023B   443             BBS     S^#QA$K_Loop_Test_Done, -        ; If done looping on this test, [25]
                              1F       0242   444             L^QA$AOB_Flags, 125$             ; branch                        [25]
                          FE32   31  0243   445             BRW     CallTest                         ; Not done looping => run again [25]
                                     0246   446
                00000000'EF   04   E1  0246   447 123$:   BBC     S^#QA$K_Run_Backwards, -         ; If the Run Backwards check is [25]
                              14       024D   448               L^QA$AOB_Check_State, 125$     ; not running, branch           [25]
                                     024E   449
    FFFFFFFF 8F   00000000'EF   F1  024E   450 124$:   ACBL    L^DS$GL_FSTTEST,   -             ; Subtract one from DSA$GL_TESTNO.[2
                  0000FE50'EF        0259
                                     025E   451             #-1,               -             ; compare that to DS$GL_FSTTEST,[25]
                          FE18       025E   452             L^DSA$GL_TESTNO, -               ; if the former is greater than [25]
                                     0260   453             CALLTEST                         ; or equal to the latter, branch.[25
                            0E   11  0260   454             BRB     130$                     ; Otherwise, we are done Run    [25]
                                     0262   455                                              ; Backwards check.              [25]
                                     0262   456         ;+                                                                   [25]
                                     0262   457         ; This is the normal way to advance through the tests.               [25]
                                     0262   458         ; Add one to the DSA$GL_TESTNO, if it is less than or equal to the   [25]
                                     0262   459         ; DS$GL_LSTTEST, branch back to CALLTEST and call the next test.     [25]
                                     0262   460         ;-                                                                   [25]
                                     0262   461
0000FE50'EF     01   00000000'EF   F1  0262   462 125$:   ACBL    L^DS$GL_LSTTEST, #1, -
                          FE08       026E   463             L^DSA$GL_TESTNO, CALLTEST        ; Do next test until reaches LSTTEST
```

```
                     0270   465          ;+
                     0270   466          ; Come to here if there are no more tests to execute in the sequence.
                     0270   467          ; Go back to FIRSTTEST and perform the initialization code again (and
                     0270   468          ; perhaps call DSX$ENDPASS in the init code).
                     0270   469          ;-
                     0270   470
                     0270   471 130$:    Br_If_Not_ExeTst        140$            ; Branch if no test executed.   [32]
                     0278   472          Clear_ExeTst                            ; Clear the execute flag        [32]
                     0280   473          QA_MAIN Dispat_Done_Tests               ; Call QA$Main.                 [31]
           FDA6   31 0289   474          BRW     FIRSTTEST                       ; GO RUN ANOTHER PASS
                     028C   475
                     028C   476          ;+
                     028C   477          ; If there were no tests executed in this section, come to here.
                     028C   478          ;-
                     028C   479
                     028C   480 140$:    $print  #ds$k_type_no_tests, -          ; No tests in section           [24]
                     028C   481                    #ds$k_printi, -               ; .. use PRINTI (was TYPMSG)    [24]
                     028C   482                    L^T_NOEXE                     ; .. 'no tests in section'      [24]
                     029F   483
                     029F   484 DISPATCH_X:
                  05 029F   485          RSB                                     ; RETURN TO COMMAND MODE
```

```
                        487          .SBTTL  DSX$EndPass Routine
                        488 ;++
                        489 ; FUNCTIONAL DESCRIPTION:
                        490 ;
                        491 ;       This routine is called to mark the end of a 'PASS'.
                        492 ;       If enough passes have been run execute user summary and cleanup code.
                        493 ;
                        494 ; CALLING SEQUENCE:
                        495 ;
                        496 ;       CALLx   #0,@#DSX$ENDPASS
                        497 ;
                        498 ; INPUT PARAMETERS:     NONE
                        499 ;
                        500 ; IMPLICIT INPUTS:      NONE
                        501 ;
                        502 ; OUTPUT PARAMETERS:    NONE
                        503 ;
                        504 ; IMPLICIT OUTPUTS:     NONE
                        505 ;
                        506 ; COMPLETION CODES:     NONE
                        507 ;
                        508 ; SIDE EFFECTS:         NONE
                        509 ;--
```

```
                          0004  02A0   511 .ENTRY  DSX$ENDPASS,^M<R2>
                                02A2   512
                                02A2   513         QA_MAIN Dispat_DSX$EndPass_Bgn  ; Call QA$Main                                [22]
                                02AB   514
                 0000FE08'EF    D5  02AB   515         TSTL    L^DSA$GL_PASSES         ; If passes = 0, => run for infinity
                           7D   13  02B1   516         BEQL    10$                     ; If = 0, branch
                                02B3   517
     0000FE54'EF 0000FE08'EF    D1  02B3   518         CMPL    L^DSA$GL_PASSES, -      ; Compare passes and passno
                                02BE   519                   L^DSA$GL_PASSNO
                           70   1A  02BE   520         BGTRU   10$                     ; If passes > passno, branch
                                02C0   521                                         ; If passno >= passes, continue on      [29]
                                02C0   522         Clear_Ctrl0                     ; Clear ^O flag                         [32]
                                02C8   523         $DS_SUMMARY_S                   ; Execute diagnostics summary section
                 0000036A'EF    16  02CF   524         JSB     DS_CLEANUP              ; Execute diagnostics cleanup section
                                02D5   525         Clear_Ctrl0                     ; Clear ^O flag                         [32]
                 00000000'EF    16  02DD   526         JSB     INIT_CONTEXT            ; Reset stacks and processor mode       [26]
                                02E3   527
                                02E3   528         $PRINT  #ds$k_type_program_end,-; Print 'run finished'                  [24]
                                02E3   529                 #ds$k_printf, -         ; .. use PRINTF                         [24]
                                02E3   530                 L^FMT[STPAS, -          ; .. format text                        [24]
                                02E3   531                 L^DS$GL_ERRCNT, -       ; .. count of error                     [24]
                                02E3   532                 L^DSA$GL_PASSNO, -      ; .. pass count                         [24]
                                02E3   533                 #0                      ; .. current time                       [24]
```

```
                          0304   535          QA_MAIN Dispat_DSX$EndPass_Mid  ; Call QA$Main                          [29]
                          030D   536          Br_If_Not_QA    5$              ; If not running QA, branch around      [32]
        00000000'EF   17  0315   537          JMP     VRRestart               ; Running QA: loop back to VRRestart    [29]
                          031B   538                                          ; for the next QA check.                [29]
                          031B   539 5$:                                       ;                                       [29]
                          031B   540          Set_CmdFlg                      ; Set command mode                      [32]
    0000FE40'EF   0A  D0  0323   541          MOVL    #APM$_DONE, -           ; INDICATE PROCESSING COMPLETE
                          032A   542                    L^DSA$GL_MSGTYP
        00000000'EF   17  032A   543          JMP     BEGIN                   ; Clean up the world and go back to CLI
                          0330   544
    0000FE54'EF   01  D1  0330   545 10$:     CMPL    #1, L^DSA$GL_PASSNO     ; IS THIS THE FIRsl PASS?
                  1B  12  0337   546          BNEQ    20$                     ; SKIP IF NOT
                          0339   547
                          0339   548          $PRINT  #ds$k_type_first_pass, -; Print first pass message       [24]
                          0339   549                    #ds$k_printf, -       ; .. use PRINTF                 [24]
                          0339   550                    L^FMTPASONE, -        ; .. format                     [24]
                          0339   551                    L^DS$GL_ERRCNT, -     ; .. error count so far         [24]
                          0339   552                    #0                    ; .. current time               [24]
                          0354   553
        0000FE54'EF   D6  0354   554 20$:     INCL    L^DSA$GL_PASSNO         ; COUNT THIS PASS
                          035A   555
                          035A   556 DSX$ENDPASS_X:
                          035A   557
                          035A   558          QA_MAIN Dispat_DSX$EndPass_End  ; Call QA$Main                          [22]
                          0363   559
        00000000'EF   16  0363   560          JSB     KB_CHECK                ; CHECK KEYBOARD STATUS.
                      04  0369   561          RET
```

```
                        563          .SBTTL  DS_Cleanup Routine
                        564 ;++
                        565 ; FUNCTIONAL DESCRIPTION:
                        566 ;
                        567 ;       EXECUTE USER'S CLEANUP CODE AND DEALLOCATE DEVICES.
                        568 ;
                        569 ; CALLING SEQUENCE:
                        570 ;
                        571 ;       BSBW    DS_CLEANUP
                        572 ;
                        573 ; INPUT PARAMETERS:     NONE
                        574 ;
                        575 ; IMPLICIT INPUTS:
                        576 ;
                        577 ;       L$A_CCP         = USER'S CLEANUP ROUTINE ADDRESS.
                        578 ;       L$L_UNIT        = MAXIMUM NUMBER OF UNITS.
                        579 ;
                        580 ; OUTPUT PARAMETERS:    NONE
                        581 ;
                        582 ; IMPLICIT OUTPUTS:     NONE
                        583 ;
                        584 ; COMPLETION CODES:     NONE
                        585 ;
                        586 ; SIDE EFFECTS:        NONE
                        587 ;--
```

```
                                 036A    589 DS_CLEANUP::
                                 036A    590         Br_If_StrFlg    1$              ; If started, continue on                [32]
                          00B7   31  0372    591         BRW     30$                     ; Exit if not started                    [22]
                                 0375    592
                                 0375    593 1$:     Br_If_Not_DonFlg        2$      ; If not done already, branch around    [32]
                          00AC   31  037D    594         BRW     30$                     ; Exit if cleanup already done           [22]
                                 0380    595
                                 0380    596 2$:     Set_DonFlg                      ; Indicate that cleanup has been done    [32]
                            07   BB  0388    597         PUSHR   #^M<R0,R1,R2>           ; SAVE REGISTERS.
                                 038A    598
                                 038A    599         QA_MAIN Dispat_DS_Cleanup_Bgn   ;                                        [22]
                                 0393    600
                                 0393    601         Clear_CtrlC                     ; Clear ^C pending bit to avoid ^C       [32]
                                 039B    602                                         ; ... on cleanup.                        [32]
                                 039B    603         $DS_CNTRLC_S                    ; DELETE USER CONTROL-C HANDLER
                                 03A6    604         $CANTIM_S                       ; CANCEL ALL TIMERS
                  0000F 50'EF    D4  03AF    605         CLRL    L^DSA$GL_TESTNO         ; TEST 0,
            0000FE4C'EF    02    D0  03B5    606         MOVL    #2, L^DSA$GL_SUBTNO     ; SUBTEST 2 INDICATES CLEANUP.
                                 03BC    607
            00000240'FF    00    FB  03BC    608         CALLS   #0, @L$A_CCP            ; EXECUTE USER'S CLEANUP.
```

```
                                        03C3   610          Br_If_User      20$                     ; Branch if user mode                   [32]
            03   00000000'EF    91      03CB   611          CmpB    L^Exe$GB_CpuType, -             ; Check if CPU is                       [34]
                                        03D2   612                   #Pr$_Sid_Typ730               ; .. a Nebula                           [34]
                           1C   12      03D2   613          BNeq    3$                              ; Skip if not                           [34]
            50   00F26200'EF    9E      03D4   614          MovAB   Ioc$K_IOSpace!^XF26200, R0       ; Address of IDC                        [34]
            51   00000004'EF    DE      03DB   615          MovAL   L^Scb_Base+4, R1                ; Get mchk handler address              [34]
                           61    DD      03E2   616          PushL   (R1)                            ; Save old one                          [34]
            61   00000000'EF    9E      03E4   617          MovAB   L^Tst$MChk, (R1)                ; Use common cheap one                  [34]
                           60    D4      03EB   618          ClrL    (R0)                            ; Clear register                        [34]
                           61  8ED0      03ED   619          PopL    (R1)                            ; Restore old handler                   [34]
                                        03F0   620
    02  00000204'EF   02    00    ED      03F0   621 3$:      CMPZV   #0,#2,L$L_ENVIRON, -
                                        03F9   622                   #SEP_FUNCTIONAL               ; SEP_FUNCTIONAL
                           07    12      03F9   623          BNEQ    5$                              ; Branch if not
            00000000'EF    6C    FA      03FB   624          CALLG   (AP), L^QIO$CLEANUP             ; CLEAN AFTER QIO IF NECESSARY
                                        0402   625
            00000000'EF    00    FB      0402   626 5$:      CALLS   #0, L^DS$INITSCB                ; Re-initialize the SCB too.
                 00000000'EF    16      0409   627          JSB     L^RMS$CLEANUP                   ; Clean up RMS storage
            50   00000000'EF    9A      040F   628          MOVZBL  L^DS$GB_MM_ENB, R0              ; Get state of memory managment
                           03    13      0416   629          BEQL    10$                             ; Branch if not enabled
                      50   01    D0      0418   630          MOVL    #1,R0                           ; Set to enable
                                        041B   631
                 00000000'EF    16      041B   632 10$:      JSB     L^DSR$MMENABLE                  ; Set MM according to R0
                                        0421   633
                                        0421   634 20$:      QA_MAIN Dispat_DS_Cleanup_End           ;                                       [22]
                           07    BA      042A   635          POPR    #^M<R0,R1,R2>                   ; RESTORE REGISTERS.                    [22]
                                        042C   636
                                 05     042C   637 30$:      RSB                                     ; RETURN.
```

```
                        639          .SBTTL  DSX$DoSummary and VRSummary Routines
042D    640 ;++
042D    641 ; FUNCTIONAL DESCRIPTION:
042D    642 ;
042D    643 ;
042D    644 ; CALLING SEQUENCE:
042D    645 ;
042D    646 ;       CALLS   #0,DS$SUMMARY   -or-
042D    647 ;       BSBW    VRSUMMARY               from command scanner
042D    648 ;
042D    649 ; INPUT PARAMETERS:     NONE
042D    650 ;
042D    651 ; IMPLICIT INPUTS:      NONE
042D    652 ;
042D    653 ; OUTPUT PARAMETERS:    NONE
042D    654 ;
042D    655 ; IMPLICIT OUTPUTS:     NONE
042D    656 ;
042D    657 ; COMPLETION CODES:     NONE
042D    658 ;
042D    659 ; SIDE EFFECTS:         NONE
042D    660 ;--
```

```
                    0000  042D   662 .ENTRY  DSX$DOSUMMARY,^M<>
00000000'EF   0E     90  042F   663         Movb    #Ds$K_Type_Summary, -          ; Set Type code to              [28]
                         0436   664                   L^Ds$GB_TypeCode            ; .. Summary                    [28]
                 71  10  0436   665         Bsbb    Display_Summary             ; Type out Summary stuff        [28]
      00000000'EF    16  0438   666         JSB     KB_CHECK
                     04  043E   667         RET
                         043F   668
                         043F   669 VRSUMMARY::
      00000000'EF    16  043F   670         Jsb     DSR$CheckLoad               ; Check to see if a program is  [36]
            4D 50    E9  0445   671         Blbc    R0, 10$                     ; . . . loaded. If not, branch  [36]
                         0448   672
                         0448   673         $Print  #-ds$k_type_summary, -      ; Summary type (sticky)         [28]
                         0448   674                   #ds$k_printf, -           ; .. use PRINTF                 [28]
                         0448   675                   L^Fmt_Summary, -          ; .. format string              [28]
                         0448   676                   L^l$a_name, -             ; .. diagnostic name            [28]
                         0448   677                   L^l$l_rev, -              ; .. revision level             [28]
                         0448   678                   L^l$l_update, -           ; .. update level               [28]
                         0448   679                   L^ds$gl_errcnt, -         ; .. error counter              [28]
                         0448   680                   L^Ds$GL_HardErr_Count, -  ; .. Hard error counter         [28]
                         0448   681                   L^Ds$GL_SoftErr_Count, -  ; .. Soft error counter         [28]
                         0448   682                   L^Ds$GL_SysErr_Count, -   ; .. System error counter       [28]
                         0448   683                   L^Ds$GL_DevErr_Count, -   ; .. Device error counter       [28]
                         0448   684                   L^Ds$GL_ErrSup_Count      ; .. ErrSup counter             [28]
                 15  10  0492   685         Bsbb    Display_Summary             ; Type out Summary stuff        [28]
                     05  0494   686         Rsb                                 ;                               [28]
```

```
                          0495    688 10$:    $Print  #ds$k_type_command_err, -       ; Command error                  [28]
                          0495    689                   #ds$k_printf, -               ; .. use PRINTF                  [28]
                          0495    690                   L^fmt_nodiag                  ; Print 'no diagnostic loaded'   [28]
                       05 04A8    691         Rsb                                   ;                                [28]
                          04A9    692
                          04A9    693 Display_Summary:                              ; Execute Summary command        [28]
          00000244'FF    00  FB 04A9    694         Calls   #0, @L$A_Repp             ; Call user report code.         [28]
              00000000'EF    94 04B0    695         Clrb    L^Ds$GB_TypeCode          ; Clear sticky type code         [28]
                       05 04B6    696         Rsb                                   ;                                [28]
```

```
                        698          .SBTTL  VRShowStatus Routine
                        699 ;++
                        700 ; Functional Description:
                        701 ;       Execute a SHOW STATUS command (print test, section, etc).
                        702 ;
                        703 ; Calling sequence:
                        704 ;
                        705 ;       BSBW    VRSHOWSTATUS
                        706 ;
                        707 ; Input parameters: none
                        708 ;
                        709 ; Implicit inputs: none
                        710 ;
                        711 ; Outputs: none
                        712 ;
                        713 ; Implicit outputs: none
                        714 ;
                        715 ; Completion codes: none
                        716 ;
                        717 ;--
```

```
                                04B7  VRShowStatus::                                            ;                                 [20]
         00000000'EF     16     04B7               Jsb     DSR$CheckLoad                        ; Check to see if a program is    [36]
               03 50     E8     04BD               Blbs    R0, 5$                               ; ... loaded. If so, skip over    [37]
                00AF     31     04C0               Brw     100$                                 ; Not loaded, print error mess.   [37]
                                04C3
          50    3C AC    D0     04C3  5$:          MovL    60(AP), R0                           ; Get the PC from the CLI args    [37]
     50   00010000 8F    D1     04C7               CmpL    #^X00010000, R0                      ; Is this a real user PC?         [37]
                  07     14     04CE               Bgtr    10$                                  ; Yes, branch around              [37]
                                04D0
       00000000'EF    00 FB     04D0               Calls   #0, L^DSR$Find_User_PC               ; Get last user PC from stack     [37]
                                04D7
 51    0000FE10'EF    01 C1     04D7  10$:         addl3   #1, dsa$gl_sectno, r1                ; Get offset in section table     [23]
   51   00000250'FF41    D0     04DF               movl    @l$a_secnam[r1], r1                  ; Get ASCIC name address          [23]
                                04E7
             50   00     D1     04E7               CmpL    #0, R0                               ; Is this really a valid PC?      [37]
                  44     13     04EA               Beql    50$                                  ; No, branch                      [37]
                                04EC
                                04EC               $print  #ds$k_type_command_out, -            ; Command output message          [24]
                                04EC                         #ds$k_printf, -                    ; .. use PRINTF                   [24]
                                04EC                         L^fmt_showstatus, -                ; .. format string                [24]
                                04EC                         l$a_name, -                        ; .. diagnostic name              [20]
                                04EC                         l$l_rev, -                         ; .. revision level               [23]
                                04EC                         l$l_update, -                      ; .. update level                 [23]
                                04EC                         #0, -                              ; .. current time                 [23]
                                04EC                         r1, -                              ; .. address of section name      [20]
                                04EC                         ds$gl_errcnt, -                    ; .. error counter                [23]
                                04EC                         dsa$gl_testno, -                   ; .. test number                  [20]
                                04EC                         dsa$gl_subtno, -                   ; .. subtest number               [20]
                                04EC                         dsa$gl_passno, -                   ; .. pass number                  [20]
                                04EC                         r0                                 ; .. User PC of ^C                [20]
                         05     052F               rsb                                          ;                                 [20]
                                0530
                                0530  50$:         $print  #ds$k_type_command_out, -            ; Command output message          [24]
                                0530                         #ds$k_printf, -                    ; .. use PRINTF                   [24]
                                0530                         L^fmt_ShowStatus_No_PC, -          ; .. format string                [24]
                                0530                         l$a_name, -                        ; .. diagnostic name              [20]
                                0530                         l$l_rev, -                         ; .. revision level               [23]
                                0530                         l$l_update, -                      ; .. update level                 [23]
                                0530                         #0, -                              ; .. current time                 [23]
                                0530                         r1, -                              ; .. address of section name      [20]
                                0530                         ds$gl_errcnt, -                    ; .. error counter                [23]
                                0530                         dsa$gl_testno, -                   ; .. test number                  [20]
                                0530                         dsa$gl_subtno, -                   ; .. subtest number               [20]
                                0530                         dsa$gl_passno                      ; .. pass number                  [37]
                         05     0571               rsb                                          ;                                 [20]
                                0572
                                0572  100$:        $print  #ds$k_type_command_err, -            ; Command error                   [24]
                                0572                         #ds$k_printf, -                    ; .. use PRINTF                   [24]
                                0572                         L^fmt_nodiag                       ; Print 'no diagnostic running'   [20]
                         05     0585               rsb                                          ;                                 [20]
                                0586               .END
```

```
$$N                             = 00000001     D
$$T1                            = 00000001     D
$ER                             = 00000002     D
$MODULE                           00000000 R   D   03
APC$_ABORT                      = 00000006     D
APC$_CONTINUE                   = 00000009     D
APC$_DUMP                       = 00000003     D
APC$_NOP                        = 00000000     D
APC$_START                      = 00000005     D
APC$_ZERO                       = 00000004     D
APM$_ABTDON                     = 00000006     D
APM$_DEVERR                     = 00000002     D
APM$_DONE                       = 0000000A     D
APM$_EXCEPT                     = 0000000B     D
APM$_HARDERR                    = 0000C003     D
APM$_MORE                       = 00000008     D
APM$_NOMES                      = 00000000     D
APM$_PREPERR                    = 0000000C     D
APM$_PRGERR                     = 00000005     D
APM$_SOFTERR                    = 00000004     D
APM$_SPOOL                      = 00000009     D
APM$_SYSERR                     = 00000001     D
A_DATAPTR                         00000004 R   D   02
A_TESTPTR                         00000000 R   D   02
BEGIN                             ********     X   04
BIT...                          = 00000004     D
CALLTEST                          00000078 R   D   04
CEP_FUNCTIONAL                  = 00000000     D
CEP_REPAIR                      = 00000001     D
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        00000444     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEST                        00000020     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
DISPATCH                          00000000 RG  D   04
DISPATCH_X                        0000029F R   D   04
DISPLAY_SUMMARY                   000004A9 R   D   04
DS$CNTRLC                         ********     X   04
DS$GA_CHKLPPC                     ********     X   04
DS$GB_MM_ENB                      ********     X   04
DS$GB_TYPECODE                    ********     X   04
DS$GL_CLIBASE                     ********     X   04
DS$GL_DEVERR_COUNT                ********     X   04
DS$GL_ERRCNT                      ********     X   04
DS$GL_ERRSUP_COUNT                ********     X   04
DS$GL_FLAGS                       ********     X   04
DS$GL_FSTTEST                     ********     X   04
DS$GL_HARDERR_COUNT               ********     X   04
DS$GL_LSTTEST                     ********     X   04
DS$GL_SOFTERR_COUNT               ********     X   04
DS$GL_SUBTEST                     ********     X   04
DS$GL_SYSERR_COUNT                ********     X   04
DS$INITSCB                        ********     X   04
DS$K_BBC                        = 0000001D   G D
DS$K_BBCC                       = 00000021   G D
DS$K_BBCCI                      = 00000023   G D
DS$K_BBCS                       = 0000001F   G D
DS$K_BBS                        = 0000001C   G D
DS$K_BBSC                       = 00000020   G D
DS$K_BBSS                       = 0000001E   G D
DS$K_BBSSI                      = 00000022   G D
DS$K_BCC_M                      = 0000001B   G D
DS$K_BCS_M                      = 0000001A   G D
DS$K_BEQLU_B                    = 00000005   G D
DS$K_BEQLU_M                    = 00000011   G D
DS$K_BEQL_B                     = 00000004   G D
DS$K_BEQL_M                     = 00000010   G D
DS$K_BERROR                     = 00000026   G D
DS$K_BGEQU_B                    = 00000007   G D
DS$K_BGEQU_M                    = 00000013   G D
```

```
DS$K_BGEQ_B                     = 00000006  G D
DS$K_BGEQ_M                     = 00000012  G D
DS$K_BGTRU_B                    = 00000009  G D
DS$K_BGTRU_M                    = 00000015  G D
DS$K_BGTR_B                     = 00000008  G D
DS$K_BGTR_M                     = 00000014  G D
DS$K_BLBC                       = 00000025  G D
DS$K_BLBS                       = 00000024  G D
DS$K_BLEQU_B                    = 00000003  G D
DS$K_BLEQU_M                    = 0000000F  G D
DS$K_BLEQ_B                     = 00000002  G D
DS$K_BLEQ_M                     = 0000000E  G D
DS$K_BLSSU_B                    = 00000001  G D
DS$K_BLSSU_M                    = 0000000D  G D
DS$K_BLSS_B                     = 00000000  G D
DS$K_BLSS_M                     = 0000000C  G D
DS$K_BNEQU_B                    = 0000000B  G D
DS$K_BNEQU_M                    = 00000017  G D
DS$K_BNEQ_B                     = 0000000A  G D
DS$K_BNEQ_M                     = 00000016  G D
DS$K_BNERROR                    = 00000027  G D
DS$K_BVC_M                      = 00000019  G D
DS$K_BVS_M                      = 00000018  G D
DS$K_DS_STARTING_LOCATION       = 00010000    D
DS$K_PRINTB                     = 00000002    D
DS$K_PRINTF                     = 00000001    D
DS$K_PRINTI                     = 00000000    D
DS$K_PRINTX                     = 00000003    D
DS$K_TYPE_ABORT_PROGRAM         = 00000014    D
DS$K_TYPE_ABORT_TEST            = 00000013    D
DS$K_TYPE_COMMAND_ERR           = 00000015    D
DS$K_TYPE_COMMAND_OUT           = 00000016    D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A    D
DS$K_TYPE_DS_PROMPT             = 00000001    D
DS$K_TYPE_DS_START              = 0000001D    D
DS$K_TYPE_ERRDEV                = 00000008    D
DS$K_TYPE_ERRHARD               = 00000006    D
DS$K_TYPE_ERROR_BODY            = 00000009    D
DS$K_TYPE_ERROR_END             = 0000000A    D
DS$K_TYPE_ERRPREP               = 0000001B    D
DS$K_TYPE_ERRSOFT               = 00000007    D
DS$K_TYPE_ERRSUP                = 00000004    D
DS$K_TYPE_ERRSYS                = 00000005    D
DS$K_TYPE_ERR_HALT              = 0000000D    D
DS$K_TYPE_EXCEPTION             = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D
DS$K_TYPE_FIRST_PASS            = 00000011    D
DS$K_TYPE_GENERAL               = 00000000    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D
DS$K_TYPE_NO_TESTS              = 00000012    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D
DS$K_TYPE_PROGRAM_END           = 00000010    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D
DS$K_TYPE_QIO_INVADP            = 00000024    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D
DS$K_TYPE_START_LIST            = 00000025    D
DS$K_TYPE_SUMMARY               = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMRON                     = 00020000    D
DS$SUMMARY                      = ********  X   04
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
```

```
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_PASSO                     = 0000001D     D
DSA$V_QA                        = 0000000F     D
DSA$V_SEARCH                    = 0000000E     D
DSA$V_TRACE                     = 0000000A     D
DSA$V_USER                      = 0000001C     D
DSP$A_BASE                        00000000     D
DSP$A_DATA                        00000008     D
DSP$A_MSG                         00000004     D
DSP$A_TEST                        0000000C     D
DSP$K_SIZE                      = 00000018     D
DSP$Q_SECTION                     00000010     D
DSQA$K_DISPAT_AFTER_INIT        = 00000014     D
DSQA$K_DISPAT_BEFORE_INIT       = 00000013     D
DSQA$K_DISPAT_CALLTEST          = 00000015     D
DSQA$K_DISPAT_DONE_TESTS        = 00000018     D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   = 00000001     D
DSQA$K_DISPAT_DSX$ENDPASS_END   = 00000002     D
DSQA$K_DISPAT_DSX$ENDPASS_MID   = 00000000     D
DSQA$K_DISPAT_DS_CLEANUP_BGN    = 00000003     D
DSQA$K_DISPAT_DS_CLEANUP_END    = 00000004     D
DSQA$K_DUMMY_T                  = 00000007     D
DSQA$K_ERROR_ERROR_BGN          = 00000006     D
DSQA$K_ERROR_ERROR_END          = 00000005     D
DSQA$K_KERNEL_INIT_CONTEXT      = 00000008     D
DSQA$K_LOOP_DSX$BGNSUB          = 00000009     D
DSQA$K_LOOP_DSX$CKLOOP          = 0000000B     D
DSQA$K_LOOP_DSX$ENDSUB          = 0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS     = 0000000E     D
DSQA$K_PARAM_DSX$GETDATA        = 0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL     = 0000000F     D
DSQA$K_PARAM_DSX$GETSTRING      = 00000010     D
DSQA$K_PARAM_DSX$GETVIELD       = 0000000D     D
DSQA$K_QA_DSX$BRANCH            = 00000011     D
DSQA$K_START_BEFORE_HEADER      = C0000017     D
DSQA$K_START_VRRESTART          = 00000012     D
DSQA$K_START_VRSTART            = 00000016     D
DSR$CHECKLOAD                     ********   X   04
DSR$FIND_USER_PC                  ********   X   04
DSR$MMENABLE                      ********   X   04
DSX$DOSUMMARY                     0000042D RG  D   04
DSX$ENDPASS                       000002A0 RG  D   04
DSX$ENDPASS_X                     0000035A R   D   04
DSX$PRINT                         ********   X   04
DS_CLEANUP                        0000036A RG  D   04
DS_ERRSUP                         ********   X   04
ENV$M_DOMAIN                    = 00000002     D
ENV$M_LEVEL                     = 00000001     D
ENV$M_SUPER                     = 000003FC     D
ENV$S_DOMAIN                    = 00000001     D
ENV$S_LEVEL                     = 00000001     D
ENV$S_SUPER                     = 00000008     D
ENV$V_DOMAIN                    = 00000001     D
ENV$V_LEVEL                     = 00000000     D
ENV$V_SUPER                     = 00000002     D
ENV$_CPU                        = 00000000     D
ENV$_FUNCTIONAL                 = 00000000     D
ENV$_REPAIR                     = 00000001     D
ENV$_SUPER                      = 00000001     D
ENV$_SYSTEM                     = 00000001     D
EXE$GB_CPUTYPE                    ********   X   04
FALSE                           = 00000000     D
FIRSTTEST                         00000032 R   D   04
FMTLSTPAS                         00000065 R   D   03
FMTPASONE                         0000002A R   D   03
FMTSEARCH                         000000B2 R   D   03
FMTTRACE                          000000EA R   D   03
FMT_NODIAG                        00000274 R   D   03
FMT_SHOWSTATUS                    000001FA R   D   03
FMT_SHOWSTATUS_NO_PC              0000018F R   D   03
FMT_SUMMARY                       000000FA R   D   03
INIT_CONTEXT                      ********     X   04
IOC$R_IOSPACE                     ********     X   04
KB_CHECK                          ********     X   04
L$A_CCP                           00000240       D
L$A_DEVP                          0000021C       D
L$A_DREG                          00000224       D
L$A_DTP                           00000218       D
L$A_ICP                           0000023C       D
L$A_LASTADR                       00000214       D
L$A_NAME                          00000208       D
L$A_REPP                          00000244       D
L$A_SECNAM                        00000250       D
L$A_STATAB                        00000248       D
L$A_TSTCNT                        00000254       D
L$L_ENVIRON                       00000204       D
L$L_ERRTYP                        0000024C       D
L$L_HEADLENGTH                    00000200       D
L$L_REV                           0000020C       D
L$L_UNIT                          00000220       D
L$L_UNUSED                        00000228       D
L$L_UPDATE                        00000210       D
OFF                             = 00000000       D
```

```
ON                              = 00000001    D
PR$_SID_TYP730                  = 00000003    D
QA$AOB_CHECK_STATE                ********  X   04
QA$AOB_FLAGS                      ********  X   04
QA$K_ABORT_NOW                  = 00000007    D
QA$K_APT_PHASE_ONE              = 0000000B    D
QA$K_APT_PHASE_TWO              = 0000000C    D
QA$K_BAD_ADDRESS                = 00000009    D
QA$K_CONTROL_C_CONT             = 0000000A    D
QA$K_DEALLOCATION               = 0000000E    D
QA$K_ERROR_PHASE_ONE            = 00000005    D
QA$K_ERROR_PHASE_THREE          = 00000007    D
QA$K_ERROR_PHASE_TWO            = 00000006    D
QA$K_FAILURE                    = 00000000    D
QA$K_FIRST_BRANCH_CODE          = 0000C000    D
QA$K_FIRST_TIME                 = 00000005    D
QA$K_INIT_CONTEXT_DONE          = 00000003    D
QA$K_LAST_BRANCH_CODE           = 00000025    D
QA$K_LOOP_ON_SUBTEST            = 00000003    D
QA$K_LOOP_ON_TEST               = 00000002    D
QA$K_LOOP_TEST_DONE             = 00000004    D
QA$K_MEMORY_MANAGE              = 0000000D    D
QA$K_MULTIPLE_PASS              = 00000001    D
QA$K_NODEF                      = 00000000    D
QA$K_NORMAL_START               = 00000000    D
QA$K_NO_HEADER                  = 00000006    D
QA$K_NUMBER_OF_CHECKS           = 0000000F    D
QA$K_NUMBER_QA_FLAGS            = 00000008    D
QA$K_NUMB_ROUTINE_STATES        = 00000004    D
QA$K_QA_DEBUG                   = 00000001    D
QA$K_QA_DONE                    = 00000002    D
QA$K_RUN_BACKWARDS              = 00000004    D
QA$K_SECTION                    = 00000008    D
QA$K_SUCCESS                    = 00000001    D
QA$MAIN                           ********    X   04
QA$V_CHECK_DONE                 = 00000003    D
QA$V_DOING_CLEANUP              = 00000002    D
QA$V_ERROR_FOUND                = 00000001    D
QA$V_INIT_DONE                  = 00000000    D
QA$W_SAVE_LAST                    ********    X   04
QIO$CLEANUP                       ********    X   04
RMS$CLEANUP                       ********    X   04
SCB_BASE                          ********    X   04
SEP_FUNCTIONAL                  = 00000002    D
SEP_REPAIR                      = 00000003    D
SIZ...                          = 00000001    D
SYS$CANTIM                        ********   GX   04
TRUE                            = 00000001    D
TST$MCHK                          ********    X   04
T_NOEXE                           00000009 R  D   03
T_TSTABO                          00000290 R  D   03
VRRESTART                         ********    X   04
VRSHOWSTATUS                      0000087F RG D   04
VRSUMMARY                         0000043F RG D   04
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	0000029B (667.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000586 (1414.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000001	767 (1)	332 (1) 373 (1) #-385 (1) #-482 (1) 533 (1) 552 (1) 684 (1) #-690 (1) 748 (1) 762 (1) #-767 (1)
$$TI	=00000001	604 (1)	604 (1)
$ER	=00000002	316 (1)	#-316 (1)
$MODULE	00000000-R	184 (1)	316 (1)
APC$_ABORT	=00000006	163 (1)	
APC$_CONTINUE	=00000009	163 (1)	
APC$_DUMP	=00000003	163 (1)	
APC$_NOP	=00000000	163 (1)	
APC$_START	=00000005	163 (1)	
APC$_ZERO	=00000004	163 (1)	
APM$_ABTDON	=00000006	163 (1)	
APM$_DEVERR	=00000002	163 (1)	
APM$_DONE	=0000000A	163 (1)	#-541 (1)
APM$_EXCEPT	=0000000B	163 (1)	
APM$_HARDERR	=00000003	163 (1)	
APM$_MORE	=00000008	163 (1)	
APM$_NOMES	=00000000	163 (1)	
APM$_PREPERR	=0000000C	163 (1)	
APM$_PRGERR	=00000005	163 (1)	
APM$_SOFTERR	=00000004	163 (1)	
APM$_SPOOL	=00000009	163 (1)	
APM$_SYSERR	=00000001	163 (1)	
A_DATAPTR	00000004-R	179 (1)	#-335 (1) 345 (1) #-394 (1) #-397 (1) #-398 (1)
A_TESTPTR	00000000-R	178 (1)	#-309 (1) 346 (1)
BEGIN	00000000-XR		543 (1)
BIT...	=00000004	169 (1)	160 (1) 165 (1) 167 (1) 168 (1) 169 (1)
CALLTEST	00000078-R	290 (1)	#-287 (1) #-445 (1) #-453 (1) #-463 (1)
CEP_FUNCTIONAL	=00000000	160 (1)	
CEP_REPAIR	=00000001	160 (1)	
CLI$K_BUFSIZ	=00000100	164 (1)	164 (1)
CLI$K_SIZE	00000444	164 (1)	
CLI$L_ADDRESS	00000018	164 (1)	
CLI$L_COMMAND	00000004	164 (1)	
CLI$L_DATA	0000001C	164 (1)	
CLI$L_FLAGS	00000000	164 (1)	
CLI$L_LAST	00000024	164 (1)	#-437 (1)
CLI$L_NEXT	00000030	164 (1)	
CLI$L_PASS	0000002C	164 (1)	
CLI$L_SUBT	00000028	164 (1)	#-432 (1)
CLI$L_TEST	00000020	164 (1)	#-435 (1)
CLI$Q_BUFQWD	00000034	164 (1)	
CLI$Q_FILE	00000008	164 (1)	
CLI$Q_SECTION	00000010	164 (1)	
CLI$Q_TIME	0000043C	164 (1)	
CLI$T_BUFFER	0000003C	164 (1)	
CLI$V_ADAPTER	=00000018	164 (1)	

```
CLI$V_ADR                       =0000000B     164     (1)
CLI$V_ASCII                     =00000013     164     (1)
CLI$V_BREAK                     =0000000A     164     (1)
CLI$V_BRIEF                     =0000001B     164     (1)
CLI$V_BYTE                      =0000000D     164     (1)
CLI$V_CLEAR                     =00000002     164     (1)
CLI$V_DEC                       =00000010     164     (1)
CLI$V_DEFAULT                   =0000000C     164     (1)
CLI$V_DEPOSIT                   =00000019     164     (1)
CLI$V_EVENT                     =00000008     164     (1)
CLI$V_EXAM                      =00000005     164     (1)
CLI$V_FLAGS                     =00000009     164     (1)
CLI$V_HEX                       =00000012     164     (1)
CLI$V_KERNEL                    =00000017     164     (1)
CLI$V_LOAD                      =00000006     164     (1)
CLI$V_LONG                      =0000000F     164     (1)
CLI$V_NOTNUF                    =00000001     164     (1)
CLI$V_OCT                       =00000011     164     (1)
CLI$V_PREG                      =0000001A     164     (1)
CLI$V_QA                        =00000007     164     (1)
CLI$V_QACKLOOPLOOPS             =0000001C     164     (1)
CLI$V_QAERRORPRINTS             =0000001B     164     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     164     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     164     (1)
CLI$V_QATESTLOOPS               =0000001D     164     (1)
CLI$V_REG                       =00000014     164     (1)
CLI$V_REQUIRED                  =00000000     164     (1)
CLI$V_RUN                       =00000015     164     (1)
CLI$V_SET                       =00000003     164     (1)
CLI$V_SHOW                      =00000004     164     (1)
CLI$V_VALSEC                    =00000016     164     (1)
CLI$V_WORD                      =0000000E     164     (1)
DISPATCH                         00000000-R   256     (1)
DISPATCH_X                       0000029F-R   484     (1)     #-317     (1)
DISPLAY_SUMMARY                  000004A9-R   693     (1)     #-665     (1)     #-685     (1)
DS$CNTRLC                        00000000-XR                    603     (1)
DS$GA_CHKLPPC                    00000000-XR                  #-337     (1)
DS$GB_MM_ENB                     00000000-XR                  #-628     (1)
DS$GB_TYPECODE                   00000000-XR                  #-664     (1)     #-695     (1)
DS$GL_CLIBASE                    00000000-XR                    431     (1)
DS$GL_DEVERR_COUNT               00000000-XR                  #-261     (1)     #-684     (1)
DS$GL_ERRCNT                     00000000-XR                  #-257     (1)     #-533     (1)     #-552     (1)     #-684     (1)
                                                              #-748     (1)     #-762     (1)
DS$GL_ERRSUP_COUNT               00000000-XR                  #-262     (1)     #-684     (1)
DS$GL_FLAGS                      00000000-XR                    297     (1)       339     (1)       353     (1)       354     (1)
                                                                361     (1)       471     (1)       472     (1)       522     (1)
                                                                525     (1)       540     (1)       590     (1)       593     (1)
                                                                596     (1)       601     (1)
DS$GL_FSTTEST                    00000000-XR                  #-280     (1)     #-363     (1)     #-436     (1)     #-450     (1)
DS$GL_HARDERR_COUNT              00000000-XR                  #-258     (1)     #-684     (1)
DS$GL_LSTTEST                    00000000-XR                  #-362     (1)     #-364     (1)     #-373     (1)     #-438     (1)
                                                              #-462     (1)
DS$GL_SOFTERR_COUNT              00000000-XR                  #-259     (1)     #-684     (1)
DS$GL_SUBTEST                    00000000-XR                  #-433     (1)
DS$GL_SYSERR_COUNT               0000U000-XR                  #-260     (1)     #-684     (1)
DS$INITSCB                       00000000-XR                    626     (1)
DS$K_BBC                        =0000001D     168     (1)
```

```
DS$K_BBCC                       =00000021       168     (1)
DS$K_BBCCI                      =00000023       168     (1)
DS$K_BBCS                       =0000001F       168     (1)
DS$K_BBS                        =0000001C       168     (1)
DS$K_BBSC                       =00000020       168     (1)
DS$K_BBSS                       =0000001E       168     (1)
DS$K_BBSSI                      =00000022       168     (1)
DS$K_BCC_M                      =00000018       168     (1)
DS$K_BCS_M                      =0000001A       168     (1)
DS$K_BEQLU_B                    =00000005       168     (1)
DS$K_BEQLU_M                    =00000011       168     (1)
DS$K_BEQL_B                     =00000004       168     (1)
DS$K_BEQL_M                     =00000010       168     (1)
DS$K_BERROR                     =00000026       168     (1)
DS$K_BGEQU_B                    =00000007       168     (1)
DS$K_BGEQU_M                    =00000013       168     (1)
DS$K_BGEQ_B                     =00000006       168     (1)
DS$K_BGEQ_M                     =00000012       168     (1)
DS$K_BGTRU_B                    =00000009       168     (1)
DS$K_BGTRU_M                    =00000015       168     (1)
DS$K_BGTR_B                     =00000008       168     (1)
DS$K_BGTR_M                     =00000014       168     (1)
DS$K_BLBC                       =00000025       168     (1)     168     (1)
DS$K_BLBS                       =00000024       168     (1)
DS$K_BLEQU_B                    =00000003       168     (1)
DS$K_BLEQU_M                    =0000000F       168     (1)
DS$K_BLEQ_B                     =00000002       168     (1)
DS$K_BLEQ_M                     =0000000E       168     (1)
DS$K_BLSSU_B                    =00000001       168     (1)
DS$K_BLSSU_M                    =0000000D       168     (1)
DS$K_BLSS_B                     =00000000       168     (1)     168     (1)
DS$K_BLSS_M                     =0000000C       168     (1)
DS$K_BNEQU_B                    =0000000B       168     (1)
DS$K_BNEQU_M                    =00000017       168     (1)
DS$K_BNEQ_B                     =0000000A       168     (1)
DS$K_BNEQ_M                     =00000016       168     (1)
DS$K_BNERROR                    =00000027       168     (1)
DS$K_BVC_M                      =00000019       168     (1)
DS$K_BVS_M                      =00000018       168     (1)
DS$K_DS_STARTING_LOCATION       =00010000       168     (1)
DS$K_PRINTB                     =00000002       169     (1)
DS$K_PRINTF                     =00000001       169     (1)     #-332   (1)     #-373   (1)     #-533   (1)     #-552   (1)
                                                                #-684   (1)     #-690   (1)     #-748   (1)     #-762   (1)
                                                                #-767   (1)
DS$K_PRINTI                     =00000000       169     (1)     #-385   (1)     #-482   (1)
DS$K_PRINTX                     =00000003       169     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014       169     (1)
DS$K_TYPE_ABORT_TEST            =00000013       169     (1)     #-385   (1)
DS$K_TYPE_COMMAND_ERR           =00000015       169     (1)     #-690   (1)     #-767   (1)
DS$K_TYPE_COMMAND_OUT           =00000016       169     (1)     #-748   (1)     #-762   (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A       169     (1)
DS$K_TYPE_DS_PROMPT             =00000001       169     (1)
DS$K_TYPE_DS_START              =0000001D       169     (1)
DS$K_TYPE_ERRDEV                =00000008       169     (1)
DS$K_TYPE_ERRHARD               =00000006       169     (1)
DS$K_TYPE_ERROR_BODY            =00000009       169     (1)
DS$K_TYPE_ERROR_END             =0000000A       169     (1)
```

```
DS$K_TYPE_ERRPREP               =0000001B      169     (1)
DS$K_TYPE_ERRSOFT               =00000007      169     (1)
DS$K_TYPE_ERRSUP                =00000004      169     (1)
DS$K_TYPE_ERRSYS                =00000005      169     (1)
DS$K_TYPE_ERR_HALT              =0000000D      169     (1)
DS$K_TYPE_EXCEPTION             =0000000C      169     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      169     (1)
DS$K_TYPE_FIRST_PASS            =00000011      169     (1)     #-552     (1)
DS$K_TYPE_GENERAL               =00000000      169     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      169     (1)
DS$K_TYPE_NO_TESTS              =00000012      169     (1)     #-482     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      169     (1)
DS$K_TYPE_PROGRAM_END           =00000010      169     (1)     #-533     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      169     (1)     #-332     (1)     #-373     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      169     (1)
DS$K_TYPE_QIO_INVADP            =00000024      169     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      169     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      169     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      169     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      169     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      169     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      169     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      169     (1)
DS$K_TYPE_START_ERR             =00000018      169     (1)
DS$K_TYPE_START_LIST            =00000025      169     (1)
DS$K_TYPE_SUMMARY               =0000000E      169     (1)     #-663     (1)     #-684     (1)
DS$K_TYPE_USER_PROMPT           =00000002      169     (1)
DS$M_ABRTFLG                    =00000040      165     (1)
DS$M_BADTIME                    =00100000      165     (1)
DS$M_BATCH                      =00400000      165     (1)
DS$M_BRKCLR                     =00001000      165     (1)
DS$M_BRKPT                      =00000800      165     (1)
DS$M_CHARFLG                    =00000100      165     (1)
DS$M_CMDFLG                     =00000080      165     (1)
DS$M_CTRLC                      =00000001      165     (1)
DS$M_CTRLO                      =00010000      165     (1)
DS$M_DEVFLG                     =00000200      165     (1)
DS$M_DISABLCC                   =01000000      165     (1)
DS$M_DONFLG                     =00002000      165     (1)
DS$M_ERRFLG                     =00000010      165     (1)
DS$M_EXCEPT                     =00080000      165     (1)
DS$M_EXETST                     =00040000      165     (1)
DS$M_HLTFLG                     =00000008      165     (1)
DS$M_LODFLG                     =00000002      165     (1)
DS$M_MEMMGT                     =00008000      165     (1)
DS$M_OUTPUT                     =00800000      165     (1)
DS$M_RUBFLG                     =00000020      165     (1)
DS$M_SCRIPT                     =00200000      165     (1)
DS$M_SETIMR                     =02000000      165     (1)
DS$M_STRFLG                     =00000004      165     (1)
DS$M_SUBT                       =00004000      165     (1)
DS$M_SYSFLG                     =00000400      165     (1)
DS$M_TIMRON                     =00020000      165     (1)
DS$SUMMARY                       00000000-XR                            523     (1)
DS$V_ABRTFLG                    =00000006      165     (1)
DS$V_BADTIME                    =00000014      165     (1)
DS$V_BATCH                      =00000016      165     (1)
```

```
DS$V_BRKCLR                     =0000000C       165     (1)
DS$V_BRKPT                      =0000000B       165     (1)
DS$V_CHARFLG                    =00000008       165     (1)
DS$V_CMDFLG                     =00000007       165     (1)     #-540   (1)
DS$V_CTRLC                      =00000000       165     (1)     #-601   (1)
DS$V_CTRLO                      =00000010       165     (1)     #-522   (1)     #-525   (1)
DS$V_DEVFLG                     =00000009       165     (1)
DS$V_DISABLCC                   =00000018       165     (1)
DS$V_DONFLG                     =0000000D       165     (1)     #-593   (1)     #-596   (1)
DS$V_ERRFLG                     =00000004       165     (1)     #-297   (1)     #-354   (1)     #-361   (1)
DS$V_EXCEPT                     =00000013       165     (1)
DS$V_EXETST                     =00000012       165     (1)     #-353   (1)     #-471   (1)     #-472   (1)
DS$V_HLTFLG                     =00000003       165     (1)
DS$V_LODFLG                     =00000001       165     (1)
DS$V_MEMMGT                     =0000000F       165     (1)
DS$V_OUTPUT                     =00000017       165     (1)
DS$V_RUBFLG                     =00000005       165     (1)
DS$V_SCRIPT                     =00000015       165     (1)
DS$V_SETIMR                     =00000019       165     (1)
DS$V_STRFLG                     =00000002       165     (1)     #-590   (1)
DS$V_SUBT                       =0000000E       165     (1)     #-339   (1)
DS$V_SYSFLG                     =0000000A       165     (1)
DS$V_TIMRON                     =00000011       165     (1)
DSA$GL_FLAGS                     0000FE00                       264     (1)     287     (1)     327     (1)     355     (1)
                                                                382     (1)     408     (1)     536     (1)     610     (1)
DSA$GL_MSGTYP                    0000FE40                       #-542   (1)
DSA$GL_PASSES                    0000FE08                       #-365   (1)     #-423   (1)     #-424   (1)     #-430   (1)
                                                                #-515   (1)     #-518   (1)
DSA$GL_PASSNO                    0000FE54                       #-263   (1)     #-288   (1)     #-365   (1)     #-519   (1)
                                                                #-533   (1)     #-545   (1)     #-554   (1)     #-748   (1)
                                                                #-762   (1)
DSA$GL_SECTNO                    0000FE10                       #-319   (1)     #-730   (1)
DSA$GL_SUBTNO                    0000FE4C                       #-275   (1)     #-285   (1)     #-338   (1)     #-606   (1)
                                                                #-748   (1)     #-762   (1)
DSA$GL_TESTNO                    0000FE50                       #-274   (1)     #-281   (1)     #-305   (1)     #-313   (1)
                                                                #-362   (1)     #-422   (1)     #-452   (1)     #-463   (1)
                                                                #-605   (1)     #-748   (1)     #-762   (1)
DSA$V_PASSO                     =0000001D                       #-264   (1)     #-286   (1)
DSA$V_QA                        =0000000F                       #-408   (1)     #-536   (1)
DSA$V_SEARCH                    =0000000E                       #-355   (1)
DSA$V_TRACE                     =0000000A                       #-327   (1)     #-382   (1)
DSA$V_USER                      =0000001C                       #-610   (1)
DSP$A_BASE                       00000000       166     (1)     #-314   (1)     #-332   (1)
DSP$A_DATA                       00000008       166     (1)     #-334   (1)
DSP$A_MSG                        00000004       166     (1)     #-332   (1)
DSP$A_TEST                       0000000C       166     (1)     #-308   (1)
DSP$K_SIZE                      =00000018       166     (1)     #-306   (1)
DSP$Q_SECTION                    00000010       166     (1)       320   (1)
DSQA$K_DISPAT_AFTER_INIT        =00000014       167     (1)     #-283   (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013       167     (1)     #-277   (1)
DSQA$K_DISPAT_CALLTEST          =00000015       167     (1)     #-296   (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018       167     (1)     #-473   (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001       167     (1)     #-513   (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002       167     (1)     #-558   (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000       167     (1)     #-535   (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003       167     (1)     #-599   (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004       167     (1)     #-634   (1)
```

```
DSQA$K_DUMMY_1                  =00000007       167     (1)
DSQA$K_ERROR_ERROR_BGN          =00000006       167     (1)
DSQA$K_ERROR_ERROR_END          =00000005       167     (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008       167     (1)
DSQA$K_LOOP_DSX$BGNSUB          =00000009       167     (1)
DSQA$K_LOOP_DSX$CKLOOP          =000C000B       167     (1)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A       167     (1)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E       167     (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C       167     (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F       167     (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010       167     (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D       167     (1)
DSQA$K_QA_DSX$BRANCH            =00000011       167     (1)
DSQA$K_START_BEFORE_HEADER      =00000017       167     (1)
DSQA$K_START_VRRESTART          =00000C12       167     (1)
DSQA$K_START_VRSTART            =00000016       167     (1)
DSR$CHECKLOAD                    00000000-XR                    670     (1)     720     (1)
DSR$FIND_USER_PC                 00000000-XR                    728     (1)
DSR$MMENABLE                     00000000-XR                    632     (1)
DSX$DOSUMMARY                    0000042D-R     662     (1)
DSX$ENDPASS                      000002A0-R     511     (1)     366     (1)
DSX$ENDPASS_X                    0000035A-R     556     (1)
DSX$PRINT                        00000000-XR                    332     (1)     373     (1)     385     (1)     482     (1)
                                                                533     (1)     552     (1)     684     (1)     690     (1)
                                                                748     (1)     762     (1)     767     (1)
DS_CLEANUP                       0000036A-R     589     (1)     524     (1)
DS_ERRSUP                        00000000-XR                    316     (1)
ENV$M_DOMAIN                    =00000002       160     (1)
ENV$M_LEVEL                     =00000001       160     (1)
ENV$M_SUPER                     =000003FC       160     (1)
ENV$S_DOMAIN                    =00000001       160     (1)
ENV$S_LEVEL                     =00000001       160     (1)
ENV$S_SUPER                     =00000008       160     (1)
ENV$V_DOMAIN                    =00000001       160     (1)     160     (1)
ENV$V_LEVEL                     =00000000       160     (1)     160     (1)
ENV$V_SUPER                     =00000002       160     (1)
ENV$_CPU                        =00000000       160     (1)     160     (1)
ENV$_FUNCTIONAL                 =00000000       160     (1)     160     (1)
ENV$_REPAIR                     =00000001       160     (1)     160     (1)
ENV$_SUPER                      =00000001       160     (1)
ENV$_SYSTEM                     =00000001       160     (1)     160     (1)
EXE$GB_CPUTYPE                   00000000-XR                   #-611    (1)
FALSE                           =00000000       168     (1)
FIRSTTEST                        00000032-R     266     (1)    #-474    (1)
FMTLSTPAS                        00000065-R     192     (1)     533     (1)
FMTPASONE                        0000002A-R     189     (1)     552     (1)
FMTSEARCH                        000000B2-R     196     (1)     373     (1)
FMTTRACE                         000000EA-R     199     (1)     332     (1)
FMT_NODIAG                       00000274-R     221     (1)     690     (1)     767     (1)
FMT_SHOWSTATUS                   000001FA-R     215     (1)     748     (1)
FMT_SHOWSTATUS_NO_PC             0000018F-R     209     (1)     762     (1)
FMT_SUMMARY                      000000FA-R     202     (1)     684     (1)
INIT_CONTEXT                     00000000-XR                    526     (1)
IOC$R_IOSPACE                    00000000-XR                    614     (1)
KB_CHECK                         00000000-XR                    560     (1)     666     (1)
L$A_CCP                          00000240                       608     (1)
L$A_DTP                          00000218                       307     (1)
```

```
L$A_ICP                         0000023C                        279     (1)
L$A_NAME                        00000208                      #-684     (1)     #-748     (1)     #-762     (1)
L$A_REPP                        00000244                        694     (1)
L$A_SECNAM                      00000250                      #-731     (1)
L$L_ENVIRON                     00000204                        621     (1)
L$L_REV                         0000020C                      #-684     (1)     #-748     (1)     #-762     (1)
L$L_UPDATE                      00000210                      #-684     (1)     #-748     (1)     #-762     (1)
OFF                            =00000000        168    (1)
ON                             =00000001        168    (1)
PR$_SID_TYP730                 =00000003                      #-612     (1)
QA$AOB_CHECK_STATE              00000000-XR                     412     (1)       442     (1)       448     (1)
QA$AOB_FLAGS                    00000000-XR                     444     (1)
QA$K_ABORT_NOW                 =00000007        168    (1)
QA$K_APT_PHASE_ONE             =0000000B        168    (1)
QA$K_APT_PHASE_TWO             =0000000C        168    (1)
QA$K_BAD_ADDRESS               =00000009        168    (1)
QA$K_CONTROL_C_CONT            =0000000A        168    (1)
QA$K_DEALLOCATION              =0000000E        168    (1)
QA$K_ERROR_PHASE_ONE           =00000005        168    (1)
QA$K_ERROR_PHASE_THREE         =00000007        168    (1)
QA$K_ERROR_PHASE_TWO           =00000006        168    (1)
QA$K_FAILURE                   =00000000        168    (1)
QA$K_FIRST_BRANCH_CODE         =00000000        168    (1)
QA$K_FIRST_TIME                =00000005        168    (1)
QA$K_INIT_CONTEXT_DONE         =00000003        168    (1)
QA$K_LAST_BRANCH_CODE          =00000025        168    (1)
QA$K_LOOP_ON_SUBTEST           =00000003        168    (1)     #-411     (1)
QA$K_LOOP_ON_TEST              =00000002        168    (1)     #-441     (1)
QA$K_LOOP_TEST_DONE            =00000004        168    (1)     #-443     (1)
QA$K_MEMORY_MANAGE             =0000000D        168    (1)
QA$K_MULTIPLE_PASS             =00000001        168    (1)
QA$K_NODEF                     =00000000        168    (1)
QA$K_NORMAL_START              =00000000        168    (1)
QA$K_NO_HEADER                 =00000006        168    (1)
QA$K_NUMBER_OF_CHECKS          =0000000F        168    (1)
QA$K_NUMBER_QA_FLAGS           =00000008        168    (1)
QA$K_NUMB_ROUTINE_STATES       =00000004        168    (1)
QA$K_QA_DEBUG                  =00000001        168    (1)
QA$K_QA_DONE                   =00000002        168    (1)
QA$K_RUN_BACKWARDS             =00000004        168    (1)     #-447     (1)
QA$K_SECTION                   =00000008        168    (1)
QA$K_SUCCESS                   =00000001        168    (1)
QA$MAIN                         00000000-XR                     277     (1)       283     (1)       296     (1)       473     (1)
                                                                513     (1)       535     (1)       558     (1)       599     (1)
                                                                634     (1)
QA$V_CHECK_DONE                =00000003        168    (1)
QA$V_DOING_CLEANUP             =00000002        168    (1)
QA$V_ERROR_FOUND               =00000001        168    (1)
QA$V_INIT_DONE                 =00000000        168    (1)
QA$W_SAVE_LAST                  00000000-XR                   #-426     (1)
QIO$CLEANUP                     00000000-XR                     624     (1)
RMS$CLEANUP                     00000000-XR                     627     (1)
SCB_BASE                        00000000-XR                     615     (1)
SEP_FUNCTIONAL                 =00000002        160    (1)   #-622     (1)
SCP_REPAIR                     =00000003        160    (1)
SIZ...                         =00000001        165    (1)     160     (1)       165     (1)
SYS$CANTIM                      00000000-XR                     604     (1)
```

```
TRUE                            =00000001      163    (1)
TST$MCHK                         00000000-XR                 617    (1)
T_NOEXE                          00000009-R     186    (1)   482    (1)
T_TSTABO                         00000290-R     224    (1)   385    (1)
VRRESTART                        00000000-XR                 537    (1)
VRSHOWSTATUS                     000004B7-R     719    (1)
VRSUMMARY                        0000043F-R     669    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANTIM_S	1	604 (1)	604 (1)
$D1_PRINT_S	1	332 (1)	332 (1) 373 (1) 385 (1) 482 (1)
			533 (1) 552 (1) 684 (1) 690 (1)
			748 (1) 762 (1) 767 (1)
$DEF	1	169 (1)	
$DEFINI	1	159 (1)	159 (1) 161 (1) 162 (1)
$DS_CNTRLC_S	1	603 (1)	603 (1)
$DS_DSADEF	5	161 (1)	161 (1)
$DS_ENVDEF	2	160 (1)	160 (1)
$DS_HDRDEF	2	162 (1)	162 (1)
$DS_QADEF	2	168 (1)	168 (1)
$DS_SUMMARY_S	1	523 (1)	523 (1)
$DS_TYPEDEF	4	169 (1)	169 (1)
$EQU	1	169 (1)	160 (1) 167 (1) 168 (1) 169 (1)
$EQULS1	1	169 (1)	160 (1) 167 (1) 168 (1) 169 (1)
$EQULST	1	160 (1)	160 (1) 167 (1) 168 (1) 169 (1)
$GBLINI	2		160 (1) 165 (1) 167 (1) 168 (1)
			169 (1)
$PRDEF	4	159 (1)	159 (1)
$PRINT	2	328 (1)	328 (1) 369 (1) 383 (1) 480 (1)
			528 (1) 548 (1) 673 (1) 688 (1)
			736 (1) 751 (1) 765 (1)
$PUSHADR	1	316 (1)	316 (1) 603 (1)
$PUSHTWO	1	604 (1)	604 (1)
$VIELD	1	160 (1)	160 (1) 165 (1)
$VIELD1	1	169 (1)	160 (1) 165 (1)
APTDEF	1	163 (1)	163 (1)
BR_IF_NOT_DONFLG	1	593 (1)	593 (1)
BR_IF_NOT_ERRFLG	1	354 (1)	354 (1)
BR_IF_NOT_EXETST	1	471 (1)	471 (1)
BR_IF_NOT_QA	1	536 (1)	536 (1)
BR_IF_NOT_SEARCH	1	355 (1)	355 (1)
BR_IF_NOT_TRACE	1	327 (1)	327 (1) 382 (1)
BR_IF_QA	1	408 (1)	408 (1)
BR_IF_STRFLG	1	590 (1)	590 (1)
BR_IF_USER	1	610 (1)	610 (1)
CLEAR_CTRLC	1	601 (1)	601 (1)
CLEAR_CTRLO	1	522 (1)	522 (1) 525 (1)
CLEAR_ERRFLG	1	297 (1)	297 (1) 361 (1)
CLEAR_EXETST	1	472 (1)	472 (1)
CLEAR_SUBT	1	339 (1)	339 (1)
CLIDEF	3	164 (1)	164 (1)
DSFDEF	3	165 (1)	165 (1)
DSPDEF	1	166 (1)	166 (1)
DSQA	2	167 (1)	167 (1)
DS_QADEFS	6	168 (1)	168 (1)
ERRSUP_S	1	316 (1)	316 (1)
MODNAM	1	184 (1)	184 (1)
QA_MAIN	1	277 (1)	277 (1) 283 (1) 296 (1) 473 (1)
			513 (1) 535 (1) 558 (1) 599 (1)

				634	(1)
SET_CMDFLG	1	540	(1)	540	(1)
SET_DONFLG	1	596	(1)	596	(1)
SET_EXETST	1	353	(1)	353	(1)
SET_PASSO	1	264	(1)	264	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.11	00:00:00.25
Command processing	142	00:00:00.75	00:00:01.92
Pass 1	885	00:00:14.14	00:00:20.55
Symbol table sort	0	00:00:00.75	00:00:00.90
Pass 2	229	00:00:02.91	00:00:03.65
Symbol table output	50	00:00:00.34	00:00:00.67
Psect synopsis output	7	00:00:00.02	00:00:00.03
Cross-reference output	119	00:00:01.76	00:00:02.70
Assembler run totals	1468	00:00:20.80	00:00:30.69

The working set limit was 1000 pages.
79721 bytes (156 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 523 non-local and 43 local symbols.
769 source lines were read in Pass 1, producing 0 object records in Pass 2.
118 pages of virtual memory were used to define 48 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	8
DRB1:[DS.WORK]DS.MLB;218	28
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	45

769 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DISPAT/UPDA=(DISPAT.UPD,DISPAT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(2) 46 DECLARATIONS
(3) 132 RL01/2 Bootstrap driver code

```
0000     1          .TITLE  DLBTDRIVR - RL01/2 BOOT DRIVER
0000     2          .IDENT  'V02-002'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1979, 1980                                                 *
0000     8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*****************************************************************************
0000    25 ;
0000    26
0000    27 ;++
0000    28 ; FACILITY:     BOOTS
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;       This module contains the bootstrap device driver for
0000    32 ;       the RL01/2  disks.
0000    33 ;
0000    34 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    35 ;
0000    36 ; AUTHOR:  Steve Beckhardt,     CREATION DATE:  31-Oct-1979
0000    37 ;  (Original author:  Charlie Franks)
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;
0000    41 ;       02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000    42 ;                               Change interface to BOOTDRIVR for purge of UBA datapath
0000    43 ;
0000    44 ;--
```

```
0000     46          .SBTTL  DECLARATIONS
0000     47 ;
0000     48 ; INCLUDE FILES:
0000     49 ;
0000     50
0000     51          $BTDDEF                         ; Boot device types
0000     52          $IODEF                          ; I/O function codes
0000     53          $RPBDEF                         ; RPB offsets
0000     54          $SSDEF                          ; Status codes
0000     55          $UBADEF                         ; UBA definitions
0000     56          $UBIDEF                         ; 11/750 UBA definitions
0000     57
0000     58 ;
0000     59 ; MACROS:
0000     60 ;
0000     61
0000     62 ;
0000     63 ; EQUATED SYMBOLS:
0000     64 ;
0000     65 ;
0000     66 ; RL11/RL02 CONTROLLER REGISTER OFFSETS
0000     67 ;
0000     68
0000     69          $DEFINI RL                      ;START OF REGISTER DEFINITIONS
0000     70
0000     71 $DEF     RL_CS           .BLKW   1       ;CONTROL STATUS REGISTER (CSR)
0002     72          _VIELD  RL_CS,0,<-              ;START OF CSR BIT DEFINITIONS
0002     73                  <DRDY,,M>,-             ; DRIVER READY
0002     74                  <FCODE,3>,-             ; FUNCTION CODE
0002     75                  <XBA,2>,-               ; BUS ADDRESS EXTENSION BITS
0002     76                  <IE,,M>,-               ; INTERRUPT ENABLE
0002     77                  <CRDY,,M>,-             ; CONTROLLER READY
0002     78                  <DS,2>,-                ; DRIVE SELECT
0002     79                  <OPI,,M>,-              ; OPERATION INCOMPLETE
0002     80                  <CRC,,M>,-              ; DATA CRC OR HEADER CRC
0002     81                  <DLT,,M>,-              ; DATA LATE OR HEADER NOT FOUND
0002     82                  <NXM,,M>,-              ; NON-EXISTENT MEMORY
0002     83                  <DE,,M>,-               ; DRIVE ERROR
0002     84                  <CE,,M>-                ; COMPOSITE ERROR
0002     85          >                               ;END CSR BIT DEFINITIONS
0002     86
0002     87 $DEF     RL_BA           .BLKW   1       ;BUS ADDRESS REGISTER (BAR)
0004     88
0004     89 $DEF     RL_DA           .BLKW   1       ;DISK ADDRESS REGISTER (DAR)
0006     90          _VIELD  RL_DA,0,<-              ;START OF DAR BIT DEFINITIONS
0006     91                  <MRK,,M>,-              ; MARK (ALWAYS 1)
0006     92                  <STS,,M>,-              ; GET STATUS
0006     93                  <,1>,-                  ; RESERVED BIT
0006     94                  <RST,,M>,-              ; RESET
0006     95                  <,12>-                  ; RESERVED BITS
0006     96          >                               ;END OF DAR BIT DEFINITIONS
0006     97
0006     98 $DEF     RL_MP           .BLKW   1       ;MULTIPURPOSE REGISTER (MPR)
0008     99          _VIELD  RL_MP,0,<-              ;START OF MPR BIT DEFINITIONS
0008    100                  <STA,3>,-               ; DRIVE STATE
0008    101                  <BH,,M>,-               ; BRUSH HOME
0008    102                  <HO,,M>,-               ; HEADS OUT
```

```
0008   103                 <CO,,M>,-                   ; COVER OPEN
0008   104                 <HS,,M>,-                   ; HEAD SELECT
0008   105                 <TYP,,M>,-                  ; DRIVE TYPE
0008   106                 <DSE,,M>,-                  ; DRIVE SELECT ERROR
0008   107                 <VC,,M>,-                   ; VOLUME CHECK
0008   108                 <WGE,,M>,-                  ; WRITE GATE ERROR
0008   109                 <SPE,,M>,-                  ; SPIN ERROR
0008   110                 <SKTO,,M>,-                 ; SEEK TIME OUT
0008   111                 <WL,,M>,-                   ; WRITE LOCK
0008   112                 <CHE,,M>,-                  ; CURRENT HEAD ERROR
0008   113                 <WDE,,M>-                   ; WRITE DATA ERROR
0008   114             >                               ;END OF MPR BIT DEFINITIONS
0008   115
0008   116             $DEFEND RL                      ;END RL11/RL02 REGISTER DEFINITIONS
0000   117
0000   118
0000   119 ;
0000   120 ; OWN STORAGE:
0000   121 ;
0000   122
0000   123 ;
0000   124 ; Boot driver table entry
0000   125 ;
0000   126
0000   127             $BOOT_DRIVER    DEVTYPE = BTD$K_DL,-    ; Device type (DL)
0000   128                             SIZE = DL_DRVSIZ,-      ; Driver size
0000   129                             ADDR = DL_DRIVER,-      ; Driver address
0000   130                             DRIVRNAME = DLNAME      ; Driver file name
```

```
                                0000   132          .SBTTL  RL01/2 Bootstrap driver code
                                0000   133
                                0000   134 ;++
                                0000   135 ;
                                0000   136 ; Inputs:
                                0000   137 ;
                                0000   138 ;       R3      - base address of adapter's register space
                                0000   139 ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                                0000   140 ;       R6      - contains C
                                0000   141 ;       R7      - address of device's CSR
                                0000   142 ;       R8      - SIZE OF TRANSFER IN BYTES
                                0000   143 ;       R9      - address of the RPB
                                0000   144 ;       R10     - starting address of transfer (byte offset in first
                                0000   145 ;                 page ORed with starting map register number)
                                0000   146 ;
                                0000   147 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                                0000   148 ;       BUF(AP) - Buffer address
                                0000   149 ;       SIZE(AP)- Size of transfer
                                0000   150 ;
                                0000   151 ; Implicit inputs:
                                0000   152 ;
                                0000   153 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                                0000   154 ;
                                0000   155 ; Outputs:
                                0000   156 ;
                                0000   157 ;       R0 - status code
                                0000   158 ;
                                0000   159 ;               SS$_NORMAL      - successful transfer
                                0000   160 ;               SS$_CTRLERR     - fatal controller error
                                0000   161 ;
                                0000   162 ;       R3 - must be preserved
                                0000   163 ;
                                0000   164 ;       This routine destroys R1, R2, R4, R5, R6. Within the
                                0000   165 ;       routine, register usage is as follows:
                                0000   166
                                0000   167 ;--
                                0000   168
                   00000004     0000   169 BUF = 4
                   00000008     0000   170 SIZE = 8
                   00000010     0000   171 FUNC = 16
                                0000   172
                                0000   173 DL_DRIVER:
                                0000   174
                                0000   175 ;
                                0000   176 ; RESET DRIVE AND WAIT FOR IT TO SPIN UP.
                                0000   177 ;
                                0000   178
                         50 D4  0000   179          CLRL    R0                              ; CLEAR R0
50  02  08  64 A9  F0         0002   180          INSV    RPB$W_UNIT(R9),#8,#2,R0 ; GET UNIT NUMBER
                 04 A7  0B  B0  0008   181          MOVW    #RL_DA_M_RST!-                  ; PUT RESET & GET STATUS IN DAR
                                000C   182                  RL_DA_M_STS!-
                                000C   183                  RL_DA_M_MRK,RL_DA(R7)
                67  50  04  A9  000C   184 10$:     BISW3   #4,R0,RL_CS(R7)                 ; EXECUTE DRIVE RESET
                       00F8  30  0010   185          BSBW    READY                           ; WAIT FOR CONTROLLER READY
                  06 A7  1D  B3  0013   186          BITW    #RL_MP_M_HO!-                   ; HEADS,BRUSHES,STATE OK?
                                0017   187                  RL_MP_M_3H!5,RL_MP(R7)  ; ... (5 = SEEK LINEAR MODE STATE)
                         F3 13  0017   188          BEQL    10$                             ; BRANCH IF NOT: WAIT FOR DRIVE TO SPIN UP
```

```
                    67   01   B3  0019   189          BITW    #RL_CS_M_DRDY,RL_CS(R7) ; IS DRIVE READY?
                         EE   13  001C   190          BEQL    10$                     ; IF NOT, BRANCH TO WAIT FOR DRIVE READY
                                 001E   191
                                 001E   192 ;
                                 001E   193 ; FIND CURRENT DISK ADDRESS, CALCULATE CYLINDER DIFFERENCE, AND SEEK
                                 001E   194 ; DESIRED CYLINDER
                                 001E   195 ;
                                 001E   196
                67   50   08    A9  001E   197 20$:     BISW3   #8,R0,RL_CS(R7)         ; EXECUTE READ HEADER
                          00E6   30  0022   198          BSBW    READY                   ; WAIT FOR CONTROLLER READY
                             03  18  0025   199          BGEQ    30$                     ; BRANCH IF NO ERROR READING HEADER
                          00C8   31  0027   200          BRW     100$                    ; OTHERWISE, BRANCH TO ERROR HANDLING
             51   06 A7   3F    AB  002A   201 30$:     BICW3   #^O77,RL_MP(R7),R1      ; GET CURRENT CYL & SURFACE
                                 002F   202
                                 C02F   203 ;
                                 002F   204 ; NOW CONVERT LOGICAL TO PHYSICAL
                                 002F   205 ;
                                 002F   206
                    55   02    C4  002F   207          MULL    #2,R5                   ; CONVERT LOGICAL BLOCKS TO SECTORS
                         56    D4  0032   208          CLRL    R6                      ; CLEAR HIGH PART OF DIVIDEND
    54   56   55   00000050 8F    7B  0034   209          EDIV    #80,R5,R6,R4            ; R6 = DESIRED CYL = LBN/(SECTORS/CYL)
                                 003D   210                                          ; R4 = REMAINING SECTORS
                         55    D4  003D   211          CLRL    R5                      ; CLEAR HIGH PART OF DIVIDEND
              54   55   54    28  7B  003F   212          EDIV    #40,R4,R5,R4            ; R5 = DESIRED SURFACE = R5/(SECT/SUR)
                                 0044   213                                          ; R4 = DESIRED SECTOR
              56   56   07    78  0044   214          ASHL    #7,R6,R6                ; SHIFT DESIRED CYLINDER INTO R6<15:7>
         56   01   06   55    F0  0048   215          INSV    R5,#6,#1,R6             ; INSERT DESIRED SURFACE BIT INTO R6<6>
                    51   56    B1  004D   216          CMPW    R6,R1                   ; IS A SEEK NEEDED?
                         2E    13  0050   217          BEQL    50$                     ; BRANCH IF NOT.
                                 0052   218
                                 0052   219 ;
                                 0052   220 ; NEED TO PERFORM A SEEK.
                                 0052   221 ;
                                 0052   222
              51   007F 8F    AA  0052   223          BICW    #^O177,R1               ; ISOLATE CURRENT CYLINDER IN R1
         52   56   007F 8F    AB  0057   224 35$:     BICW3   #^O177,R6,R2            ; ISOLATE DESIRED CYLINDER IN R2
                    51   52    A2  005D   225          SUBW    R2,R1                   ; SUBTRACT DESIRED FROM ACTUAL
                         08    13  0060   226          BEQL    40$                     ; BRANCH IF CURRENT = DESIRED CYLINDER
                         06    1E  0062   227          BCC     40$                     ; BRANCH IF CURRENT > DESIRED CYLINDER
                    51   51    AE  0064   228          MNEGW   R1,R1                   ; ACTUAL<DESIRED, MAKE POSITIVE DIFF
                    51   04    A8  0067   229          BISW    #4,R1                   ; SET SIGN FOR MOVE TO CENTER OF DISK
         51   01   04   55    F0  006A   230 40$:     INSV    R5,#4,#1,R1             ; INSERT SURFACE BIT IN R1<4>
   04 A7      51   01    A9  006F   231          BISW3   #RL_DA_M_MRK,R1,-       ; SET MARKER AND LOAD DIFFERENCE
                                 0074   232                  RL_DA(R7)               ; WORD.
              67   50   06    A9  0074   233          BISW3   #6,R0,RL_CS(R7)         ; EXECUTE SEEK FUNCTION
                        0090   30  0078   234          BSBW    READY                   ; WAIT FOR CONTROLLER READY OR ERROR
                           03  18  0078   235          BGEQ    50$                     ; BRANCH IF NO ERROR DURING SEEK
                        0072   31  007D   236          BRW     100$                    ; OTHERWISE, BRANCH DUE TO ERROR
                                 0080   237
                                 0080   238 ;
                                 0080   239 ; SEEK, IF ANY, IS COMPLETE. EXECUTE TRANSFER FUNCTION
                                 0080   240 ;
                                 0080   241
         56   06   00    54    F0  0080   242 50$:     INSV    R4,#0,#6,R6             ; MERGE SECTOR WITH CYLINDER AND SURFACE
                                 0085   243                                          ; CYL=R6<15:7> SUR=R6<6> SEC=R6<5:0>
              04 A7      56    B0  0085   244          MOVW    R6,RL_DA(R7)            ; SET DESIRED DISK ADDRESS
              02 A7      5A    B0  0089   245          MOVW    R10,RL_BA(R7)           ; SET BUFFER ADDRESS
```

```
                52   58   B0   008D   246          MOVW    R8,R2                   ; GET WORKING COPY OF BYTES LEFT TO XFER
                                0090   247                                          ; AND ASSUME THIS IS LAST TRANSFER
             51   28   54   A3   0090   248          SUBW3   R4,#40,R1               ; R1 = SECTORS LEFT ON SURFACE
             51   0100 8F   A4   0094   249          MULW    #256,R1                 ; CONVERT TO BYTES LEFT ON SURFACE
                  51   52   B1   0099   250          CMPW    R2,R1                   ; ARE ADDITIONAL TRANSFERS REQUIRED?
                       03   1B   009C   251          BLEQU   60$                     ; BRANCH IF ANSWER IS NO.
                  52   51   B0   009E   252          MOVW    R1,R2                   ; SET BYTE COUNT FOR THIS TRANSFER
                  52   02   A6   00A1   253 60$:     DIVW    #2,R2                   ; CALCULATE TRANSFER WORD COUNT
          0C A7   52   AE   00A4   254          MNEGW   R2,RL_MP(R7)            ; SET NEG TRANSFER WORD COUNT
                  51   0C   B0   00A8   255          MOVW    #^XC,R1                 ; ASSUME READ FUNCTION
          20   10 AC   D1   00AB   256          CMPL    FUNC(AP),#IO$_WRITELBLK ; IS IT A WRITE FUNCTION?
                       03   12   00AF   257          BNEQ    70$                     ; BRANCH IF NOT
                  51   0A   B0   00B1   258          MOVW    #^XA,R1                 ; SET WRITE FUNCTION CODE
          67      51   50   A9   00B4   259 70$:     BISW3   R0,R1,RL_CS(R7)         ; MERGE UNIT # WITH FUNCTION AND EXECUTE
                     0050   30   00B8   260          BSBW    READY                   ; WAIT FOR CONTROLLER READY OR ERROR
                       35   19   00BB   261          BLSS    100$                    ; BRANCH ON ERROR
                  52   02   A4   00BD   262 80$:     MULW    #2,R2                   ; FIND BYTES TRANSFERRED THIS TIME
                  58   52   A2   00C0   263          SUBW    R2,R8                   ; UPDATE BYTES LEFT TO TRANSFER
                       1C   13   00C3   264          BEQL    90$                     ; BRANCH IF TRANSFER IS COMPLETE
                                00C5   265
                                00C5   266 ;
                                00C5   267 ; UPDATE PARAMETERS FOR NEXT TRANSFER
                                00C5   268 ;
                                00C5   269
    51   04 A7   007F 8F   AB   00C5   270          BICW3   #^0177,RL_DA(R7),R1     ; UPDATE CURRENT CYLINDER IN R1<15:7>
    56      04 A7   3F   A9   00CC   271          BISW3   #^077,RL_DA(R7),R6      ; SET SECTOR BITS TO 1'S AND -
                  56   B6   00D1   272          INCW    R6                      ; UPDATE DESIRED DISK ADDRESS IN R6
 55   56   01   06   EF   00D3   273          EXTZV   #6,#1,R6,R5             ; UPDATE DESIRED SURFACE IN R5
                  54   D4   00D8   274          CLRL    R4                      ; UPDATE DESIRED SECTOR IN R4
       5A   02 A7   B0   00DA   275          MOVW    RL_BA(R7),R10           ; UPDATE DESIRED BUFFER ADDRESS
             FF76   31   00DD   276          BRW     35$                     ; LOOP FOR NEXT TRANSFER
                                00E1   277
                                00E1   278 ;
                                00E1   279 ; TRANSFER COMPLETE - RETURN
                                00E1   280 ;
                                00E1   281
        51   08 AC   3C   00E1   282 90$:     MOVZWL  SIZE(AP),R1             ; SET TOTAL BYTES TRANSFERRED
                       07   12   00E5   283          BNEQ    95$                     ; BRANCH IF ORIGINAL SIZE WAS TRANSFERRED
    51   00008000 8F   D0   00E7   284          MOVL    #^X8000,R1              ; ELSE SIZE WAS FORCED TO 64K
            50   01   3C   00EE   285 95$:     MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                       05   00F1   286          RSB                             ; AND RETURN
                                00F2   287
                                00F2   288 ;
                                00F2   289 ; RETRY ERROR
                                00F2   290 ;
                                00F2   291
        58   08 AC   3C   00F2   292 100$:    MOVZWL  SIZE(AP),R8             ; RESTORE BYTE SIZE OF TRANSFER IN R8
                       07   12   00F6   293          BNEQ    110$                    ; BRANCH IF SIZE WAS LEGAL
    58   00001F40 8F   D0   00F8   294          MOVL    #8000,R8                ; ELSE FORCE TO 64K SIZE
5A   04 AC      09   00   EF   00FF   295 110$:    EXTZV   #0,#9,BUF(AP),R10       ; RESTORE BYTE OFFSET IN R10
            50   0054 8F   3C   0105   296          MOVZWL  #SS$_CTRLERR,R0         ; SET FATAL CONTROLLER ERROR
                       05   010A   297          RSB                             ; AND ATTEMPT RETRY
                                0108   298
                                0108   299
                                0108   300 ;
                                0108   301 ; SUBROUTINE TO WAIT FOR CONTROLLER READY OR ERROR
                                0108   302 ;
```

```
                                    0108   303
                                    0108   304 READY:                                      ;
                 67   8080 8F   B3  0108   305          BITW    #^X8080,(R7)              ;CONTROLLER READY OR ERROR?
                           F9   13  010C   306          BEQL    READY                     ;IF EQL NO
                                05  0112   307          RSB                               ;
                                    0113   308
58 45 2E 52 45 56 49 52 44 4C 44 00' 0113   309 DLNAME: .ASCIC  /DLDRIVER.EXE/            ; Driver file name
                                45  011F
                                0C  0113
                                    0120   310
                           00000120 0120   311 DL_DRVSIZ=.-DL_DRIVER
                                    0120   312
                                    0120   313          .END
```

```
$TABLE                          = 00000000 R  D  02
BTD$K_DL                        = 00000002    D
BUF                             = 00000004    D
DLNAME                            00000113 R  D  03
DL_DRIVER                         00000000 R  D  03
DL_DRVSIZ                       = 00000120    D
FUNC                            = 00000010    D
IO$_WRITELBLK                   = 00000020    D
READY                             0000010B R  D  03
RL_BA                             00000002    D
RL_CS                             00000000    D
RL_CS_M_DRDY                    = 00000001    D
RL_DA                             00000004    D
RL_DA_M_MRK                     = 00000001    D
RL_DA_M_RST                     = 00000008    D
RL_DA_M_STS                     = 00000002    D
RL_MP                             00000006    D
RL_MP_M_BH                      = 00000008    D
RL_MP_M_HO                      = 00000010    D
RPB$W_UNIT                      = 00000064    D
SIZ...                          = 00000001    D
SIZE                            = 00000008    D
SS$_CTRLERR                     = 00000054    D
SS$_NORMAL                      = 00000001    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000008 (8.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000120 (288.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	130 (2)	130 (2)
BTD$K_DL	=00000002		130 (2)
BUF	=00000004	169 (3)	295 (3)
DLNAME	00000113-R	309 (3)	130 (2)
DL_DRIVER	00000000-R	173 (3)	130 (2) 311 (3)
DL_DRVSIZ	=00000120	311 (3)	130 (2)
FUNC	=00000010	171 (3)	#-256 (3)
IO$_WRITELBLK	=00000C20		#-256 (3)
READY	0000010B-R	304 (3)	#-185 (3) #-198 (3) #-234 (3) #-260 (3)
			#-306 (3)
RL_BA	00000002		#-245 (3) #-275 (3)
RL_CS	00000000		#-184 (3) #-189 (3) #-197 (3) #-233 (3)
			#-259 (3)
RL_CS_M_DRDY	=00000001		#-189 (3)
RL_DA	00000004		#-183 (3) #-232 (3) #-244 (3) #-270 (3)
			#-271 (3)
RL_DA_M_MRK	=00000001		#-183 (3) #-231 (3)
RL_DA_M_RST	=00000008		#-181 (3)
RL_DA_M_STS	=00000002		#-182 (3)
RL_MP	00000006		#-187 (3) #-201 (3) #-254 (3)
RL_MP_M_BH	=00000008		#-187 (3)
RL_MP_M_HO	=00000010		#-186 (3)
RPB$W_UNIT	=00000064		#-180 (3)
SIZE	=00000008	170 (3)	#-282 (3) #-292 (3)
SS$_CTRLERR	=00000054		#-296 (3)
SS$_NORMAL	=00000001		#-285 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	127 (2)	127 (2)
$BTDDEF	1	51 (2)	51 (2)
$DEFINI	1	51 (2)	51 (2) 52 (2) 53 (2) 54 (2) 55 (2) 56 (2) 69 (2)
$IODEF	17	52 (2)	52 (2)
$RPBDEF	5	53 (2)	53 (2)
$SSDEF	21	54 (2)	54 (2)
$UBADEF	6	55 (2)	55 (2)
$UBIDEF	2	56 (2)	56 (2)

Performance Indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.11	00:00:00.23
Command processing	140	00:00:00.67	00:00:01.03
Pass 1	384	00:00:11.11	00:00:17.44
Symbol table sort	0	00:00:01.76	00:00:02.13
Pass 2	86	00:00:01.81	00:00:02.64
Symbol table output	5	00:00:00.06	00:00:00.06
Psect synopsis output	6	00:00:00.02	00:00:00.03
Cross-reference output	17	00:00:00.16	00:00:00.20
Assembler run totals	674	00:00:15.73	00:00:24.38

The working set limit was 1000 pages.
57693 bytes (113 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1095 non-local and 13 local symbols.
313 source lines were read in Pass 1, producing 0 object records in Pass 2.
40 pages of virtual memory were used to define 15 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	11

1135 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DLBTDRIVR/UPDA=(DLBTDRIVR.UPD,DLBTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(2) 46 DECLARATIONS
(3) 183 RK06/7 Bootstrap driver code
(4) 329 FCC - PERFORM ECC ERROR CORRECTION

```
0000     1          .TITLE  DMBTDRIVR - RK06/7 BOOT DRIVER
0000     2          .IDENT  'V02-002'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1979, 1980                                                 *
0000     8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*****************************************************************************
0000    25 ;
0000    26
0000    27 ;++
0000    28 ; FACILITY:     BOOTS
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;       This module contains the bootstrap device driver for the
0000    32 ;       RK06/7 disks.
0000    33 ;
0000    34 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    35 ;
0000    36 ; AUTHOR:  Steve Beckhardt,     CREATION DATE:  1-Nov-1979
0000    37 ;  (Original author:  Carol Peters)
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;
0000    41 ;       02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000    42 ;                               Change interface to BOOTDRIVR for purge of UBA datapath
0000    43 ;
0000    44 ;--
```

```
                                        .SBTTL  DECLARATIONS
                                ;
                                ; INCLUDE FILES:
                                ;

                                        $BTDDEF                         ; Boot device types
                                        $IODEF                          ; I/O function codes
                                        $PRDEF                          ; Processor registers
                                        $RPBDEF                         ; RPB offsets
                                        $SSDEF                          ; Status codes
                                        $UBADEF                         ; UBA definitions
                                        $UBIDEF                         ; 11/750 UBA definitions

                                ;
                                ; MACROS:
                                ;

                                ;
                                ; EQUATED SYMBOLS:
                                ;
                                ;
                                ; RK611/RK06 CONTROLLER REGISTER OFFSETS
                                ;

                                        $DEFINI RK

                        $DEF    RK_CS1          .BLKW   1       ;CONTROL STATUS REGISTER 1
                                _VIELD  RK_CS1,0,<-             ; CONTROL STATUS REGISTER 1 FIELD DEFINITION
                                        <GO,,M>,-               ; GO BIT
                                        <FCODE,4>,-             ; FUNCTION CODE
                                        <DPPE,,M>,-             ; DATA PATH PURGE ERROR
                                        <IE,,M>,-               ; INTERRUPT ENABLE
                                        <RDY,,M>,-              ; CONTROLLER READY
                                        <MEX,2>,-               ; MEMORY EXTENSION BITS
                                        <CDT,,M>,-              ; CONTROLLER DRIVE TYPE
                                        <CTO,,M>,-              ; CONTROLLER TIME OUT
                                        <CFMT,,M>,-             ; CONTROLLER FORMAT ERROR
                                        <SPAR,,M>,-             ; SERIAL BUS PARITY ERROR
                                        <DI,,M>,-               ; DRIVE INTERRUPT
                                        <CERR,,M>-              ; CONTROLLER ERROR
                                ;
                                >
                        $DEF    RK_WC           .BLKW   1       ;WORD COUNT REGISTER
                        $DEF    RK_BA           .BLKW   1       ;BUFFER ADDRESS REGISTER
                        $DEF    RK_DA           .BLKW   1       ;DESIRED SECTOR/TRACK ADDRESS REGISTER
                                _VIELD  RK_DA,0,<-              ; DESIRED ADDRESS FIELD DEFINITIONS
                                        <SA,5>,-                ; DESIRED SECTOR ADDRESS
                                        <,3>,-                  ; RESERVED BITS
                                        <TA,3>-                 ; DESIRED TRACK ADDRESS
                                ;
                                >
                        $DEF    RK_CS2          .BLKW   1       ;CONTROL STATUS REGISTER 2
                                _VIELD  RK_CS2,0,<-             ; CONTROL STATUS REGISTER 2 FIELD DEFINITION
                                        <DS,3>,-                ; DRIVE SELECT
                                        <RLS,,M>,-              ; RELEASE DRIVE
                                        <BAI,,M>,-              ; BUFFER ADDRESS INCREMENT INHIBIT
                                        <SCLR,,M>,-             ; SUBSYSTEM CLEAR
                                        <IR,,M>,-               ; INPUT READY
                                        <OR,,M>,-               ; OUTPUT READY
```

```
                     000A  103                     <UFE,,M>,-              ; UNIT FIELD ERROR
                     000A  104                     <MDS,,M>,-              ; MULTIPLE DRIVE SELECT
                     000A  105                     <PGE,,M>,-              ; PROGRAMMING ERROR
                     000A  106                     <NEM,,M>,-              ; NONEXISTENT MEMORY
                     000A  107                     <NED,,M>,-              ; NONEXISTENT DRIVE
                     000A  108                     <UPE,,M>,-              ; UNIBUS PARITY ERROR
                     000A  109                     <WCE,,M>,-              ; WRITE CHECK ERROR
                     000A  110                     <DLT,,M>-               ; DATA LATE ERROR
                     000A  111             >                               ;
                     000A  112 $DEF        RK_DS           .BLKW   1       ;DRIVE STATUS REGISTER
                     000C  113             _VIELD  RK_DS,0,<-              ; DRIVE STATUS REGISTER BIT DEFINTIONS
                     000C  114                     <DRA,,M>,-              ; DRIVE AVAILABLE
                     000C  115                     <,1>,-                  ; RESERVED BIT
                     000C  116                     <OFST,,M>,-             ; DRIVE OFFSET
                     C00C  117                     <ACLO,,M>,-             ; DRIVE AC LOW
                     000C  118                     <DCLO,,M>,-             ; DRIVE DC LOW
                     000C  119                     <DROT,,M>,-             ; DRIVE OFF TRACK
                     000C  120                     <VV,,M>,-               ; VOLUME VALID
                     000C  121                     <DRDY,,M>,-             ; DRIVE READY
                     000C  122                     <DDT,,M>,-              ; DRIVE DRIVE TYPE
                     000C  123                     <,2>,-                  ; RESERVED BITS
                     000C  124                     <WRL,,M>,-              ; DRIVE WRITE LOCKED
                     000C  125                     <,1>,-                  ; RESERVED BIT
                     000C  126                     <PIP,,M>,-              ; POSITIONING IN PROGRESS
                     000C  127                     <DSC,,M>,-              ; DRIVE STATUS CHANGE
                     000C  128                     <SVAL,,M>-              ; DRIVE STATUS VALID
                     000C  129             >                               ;
                     000C  130 $DEF        RK_ER           .BLKW   1       ;ERROR REGISTER
                     000E  131             _VIELD  RK_ER,0,<-              ; ERROR REGISTER BIT DEFINITIONS
                     000E  132                     <ILF,,M>,-              ; ILLEGAL FUNCTION
                     000E  133                     <SKI,,M>,-              ; SEEK INCOMPLETE
                     000E  134                     <NXF,,M>,-              ; NONEXECUTABLE FUNCTION
                     000E  135                     <DRPAR,,M>,-            ; DRIVE PARITY ERROR
                     000E  136                     <FMTE,,M>,-             ; FORMAT ERROR
                     000E  137                     <DTYE,,M>,-             ; DRIVE TYPE ERROR
                     000E  138                     <ECH,,M>,-              ; ECC HARD ERROR
                     000E  139                     <BSE,,M>,-              ; BAD SECTOR ERROR
                     000E  140                     <HVRC,,M>,-             ; HEADER VRC ERROR
                     000E  141                     <COE,,M>,-              ; CYLINDER OVERFLOW ERROR
                     000E  142                     <IDAE,,M>,-             ; INVALID DISK ADDRESS ERROR
                     000E  143                     <WLE,,M>,-              ; WRITE LOCK ERROR
                     000E  144                     <DTE,,M>,-              ; DRIVE TIMING ERROR
                     000E  145                     <OPI,,M>,-              ; OPERATION INCOMPLETE
                     000E  146                     <UNS,,M>,-              ; DRIVE UNSAFE
                     000E  147                     <DCK,,M>-               ; DATA CHECK ERROR
                     000E  148             >                               ;
                     000E  149 $DEF        RK_AS           .BLKW   1       ;ATTENTION SUMMARY/OFFSET REGISTER
                     0010  150             _VIELD  RK_AS,0,<-              ; ATTENTION SUMMARY/OFFSET REGISTER FIELDS
                     0010  151                     <OF,7>,-                ; DRIVE OFFSET
                     0010  152                     <,1>,-                  ; RESERVED BIT
                     0010  153                     <ATTN,8,M>-             ; DRIVE ATTENTION SUMMARY
                     0010  154             >                               ;
                     0010  155 $DEF        RK_DC           .BLKW   1       ;DESIRED CYLINDER ADDRESS
                     0012  156 $DEF        RK_SPR          .BLKW   1       ;UNUSED REGISTER
                     0014  157 $DEF        RK_DB           .BLKW   1       ;DATA BUFFER REGISTER
                     0016  158 $DEF        RK_MR1          .BLKW   1       ;MAINTENANCE REGISTER 1
                     0018  159             _VIELD  RK_MR1,0,<<MS,3>>       ; MAINTENANCE REGISTER 1 FIELD DEFINITION
```

```
0018   160 $DEF    RK_EC1          .BLKW   1       ;ECC POSITION REGISTER
001A   161          VIELD   RK_EC1,0,<<EPS,13>>    ; ECC POSITION FIELD
001A   162 $DEF    RK_EC2          .BLKW   1       ;ECC PATTERN REGISTER
001C   163          VIELD   RK_EC2,0,<<EPT,11>>    ; ECC PATTERN FIELD
001C   164 $DEF    RK_MR2          .BLKW   1       ;MAINTENANCE REGISTER 2
001E   165 $DEF    RK_MR3          .BLKW   1       ;MAINTENANCE REGISTER 3
0020   166
0020   167          $DEFEND RK
0000   168
0000   169
0000   170 ;
0000   171 ; OWN STORAGE:
0000   172 ;
0000   173
0000   174 ;
0000   175 ; Boot driver table entry
0000   176 ;
0000   177
0000   178          $BOOT_DRIVER    DEVTYPE = BTD$K_DM,-    ; Device type (DM)
0000   179                          SIZE = DM_DRVSIZ,-      ; Driver size
0000   180                          ADDR = DM_DRIVER,-      ; Driver address
0000   181                          DRIVRNAME = DMNAME      ; Driver name
```

```
                     0000   183          .SBTTL  RK06/7 Bootstrap driver code
                     0000   184
                     0000   185 ;++
                     0000   186 ;
                     0000   187 ; Inputs:
                     0000   188 ;
                     0000   189 ;       R3      - base address of adapter's register space
                     0000   190 ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                     0000   191 ;       R6      - contains 0
                     0000   192 ;       R7      - address of the device's CSR
                     0000   193 ;       R8      - SIZE OF TRANSFER IN BYTES
                     0000   194 ;       R9      - address of the RPB
                     0000   195 ;       R10     - starting address of transfer (byte offset in first
                     0000   196 ;                 page ORed with starting map register number)
                     0000   197 ;       R11     - LBN at start of transfer
                     0000   198 ;
                     0000   199 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                     0000   200 ;       BUF(AP) - Buffer address
                     0000   201 ;       SIZE(AP)- Size of transfer in bytes
                     0000   202 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                     0000   203 ;
                     0000   204 ; Implicit inputs:
                     0000   205 ;
                     0000   206 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                     0000   207 ;
                     0000   208 ; Outputs:
                     0000   209 ;
                     0000   210 ;       R0 - status code
                     0000   211 ;
                     0000   212 ;               SS$_NORMAL      - successful transfer
                     0000   213 ;               SS$_NOSUCHDEV   - unsupported device
                     0000   214 ;               SS$_CTRLERR     - fatal controller error
                     0000   215 ;
                     0000   216 ;       R3 - must be preserved
                     0000   217 ;
                     0000   218 ;       This routine destroys R1, R2, R4, R5, R6. Within the
                     0000   219 ;       routine, register usage is as follows:
                     0000   220 ;
                     0000   221 ;               R0      - mapping enabled flag
                     0000   222 ;                         device unit number
                     0000   223 ;                         status code
                     0000   224 ;               R1      - device function code
                     0000   225 ;               R2      - drive type according to device register
                     0000   226 ;               R4      - used in logical to physical calculation
                     0000   227 ;               R5      - used in logical to physical calculation
                     0000   228 ;               R6      - used in logical to physical calculation
                     0000   229 ;                         transfer word count
                     0000   230 ;
                     0000   231 ;--
00000004             0000   232 BUF = 4
00000008             0000   233 SIZE = 8
0000000C             0000   234 LBN = 12
00000010             0000   235 FUNC = 16
00000014             0000   236 MODE = 20
                     0000   237
                     0000   238 DM_DRIVER:                           ; RK06/7 device driver.
                     0000   239
```

```
                                0000   240 ;
                                0000   241 ; Translate the I/O function code into a device-dependent function
                                0000   242 ; code for this disk.
                                0000   243 ;
                                0000   244
      51    11    D0            0000   245 10$:    MOVL    #17,R1                      ; ASSUME READ
  20   10 AC    D1              0003   246          CMPL    FUNC(AP),#IO$_WRITELBLK     ; CHECK FOR WRITE FUNCTION
         03    12               0007   247          BNEQ    20$                         ; NO, DO READ
      51    13    D0            0009   248          MOVL    #19,R1                      ; SET WRITE FUNCTION CODE
                                000C   249
                                000C   250 ;
                                000C   251 ; Clear controller and drive status. Confirm that the drive exists and
                                000C   252 ; is ready for a transfer.
                                000C   253 ;
                                000C   254
   50   64 A9    3C            000C   255 20$:    MOVZWL  RPB$W_UNIT(R9),R0           ; GET UNIT NUMBER
   08 A7   20    B0            0010   256          MOVW    #RK_CS2_M_SCLR,RK_CS2(R7)   ; CLEAR CONTROLLER AND ALL DRIVES
        0092    30              0014   257          BSBW    READY                       ; WAIT FOR CONTROLLER READY
   08 A7   50    B0            0017   258          MOVW    R0,RK_CS2(R7)               ; SET DRIVE NUMBER
                                001B   259
                                001B   260 30$:                                       ; Clear drive and find type.
      67    05    B0            001B   261          MOVW    #5,RK_CS1(R7)               ; Clear drive.
         0088    30              001E   262          BSBW    READY                       ; Wait for controller ready.
            52    D4            0021   263          CLRL    R2                          ; Assume RK06 drive.
0A A7   0100 8F    B3            0023   264          BITW    #RK_DS_M_DDT,RK_DS(R7)      ; Is drive RK07?
         05    13               0029   265          BEQL    33$                         ; No. Branch.
   52   0400 8F    3C            002B   266          MOVZWL  #RK_CS1_M_CDT,R2            ; Yes. Set drive code.
                                0030   267
                                0030   268 33$:                                       ; Check for existence of drive.
08 A7   1000 8F    B3            0030   269          BITW    #RK_CS2_M_NED,RK_CS2(R7)    ; Does drive exist?
         06    13               0036   270          BEQL    35$                         ; Yes. Branch.
   50   0908 8F    3C            0038   271          MOVZWL  #SS$_NOSUCHDEV,R0           ; No.  Exit with error
         05    003D             272          RSB
                                003E   273
                                003E   274 35$:                                       ; Clear drive and acknowledge.
      8000 8F    B0              003E   275          MOVW    #RK_CS1_M_CERR,-            ; Clear controller error
            67                  0042   276                  RK_CS1(R7)                  ; status.
   08 A7   50    B0            0043   277          MOVW    R0,RK_CS2(R7)               ; Set unit number code again.
67    52   05    A9              0047   278          BISW3   #5,R2,RK_CS1(R7)            ; Clear drive.
        005B    30              004B   279          BSBW    READY                       ; Wait for controller ready.
0A A7   0080 8F    B3            004E   280          BITW    #RK_DS_M_DRDY,RK_DS(R7)     ; Is the drive ready?
         E8    13               0054   281          BEQL    35$                         ; No. Clear drive again.
67    52   03    A9              0056   282          BISW3   #3,R2,RK_CS1(R7)           ; Acknowledge pack and set
                                005A   283                                              ; volume valid.
         4D    10               005A   284          BSBB    READY                       ; Wait for controller ready.
                                005C   285
                                005C   286 ;
                                005C   287 ; Compute the cylinder, track, and sector addresses. Load device
                                005C   288 ; registers with transfer parameters and start transfer.
                                005C   289 ;
                                005C   290
            56    D4            005C   291          CLRL    R6                          ; Clear register for EDIV.
55   54   55  00000042 8F    7B  005E   292          EDIV    #22*3,R5,R4,R5              ; COMPUTE DESIRED CYLINDER
      10 A7   54    B0            0067   293          MOVW    R4,RK_DC(R7)                ; AND SET IN DEVICE
   56   55   55    16    7B      006B   294          EDIV    #22,R5,R5,R6                ; CALCULATE DESIRED TRACK/SECTOR
   56   08   08    55    F0      0070   295          INSV    R5,#8,#8,R6                 ; MERGE TRACK AND SECTOR
      06 A7   56    B0            0075   296          MOVW    R6,RK_DA(R7)                ; SET DESIRED TRACK SECTOR
```

```
                04 A7   5A   B0  0079   297          MOVW    R10,RK_BA(R7)           ; SET STARTING BUFFER ADDRESS
              56   58   02   C7  007D   298          DIVL3   #2,R8,R6                ; COMPUTE WORD COUNT
                02 A7   56   AE  0081   299          MNEGW   R6,RK_WC(R7)            ; SET NUMBER OF WORDS TO TRANSFER
              67   52   51   A9  0085   300          BISW3   R1,R2,RK_CS1(R7)        ; Start disk function.
                        1E   10  0089   301          BSBB    READY                   ; WAIT FOR CONTROLLER READY
                             008B   302
                             008B   303 ;
                             008B   304 ; Transfer is complete. See whether the transfer completed without
                             008B   305 ; error. If not, prepare input registers for the ECC correction routine
                             008B   306 ; and branch to that routine.
                             008B   307 ;
                             008B   308
                   50   01   3C  008B   309          MOVZWL  #SS$_NORMAL,R0          ; ASSUME NORMAL COMPLETION
                        67   B5  008E   310          TSTW    RK_CS1(R7)              ; CHECK COMPOSITE ERROR
                        01   19  0090   311          BLSS    50$                     ; CONTINUE IF NO ERROR
                             05  0092   312          RSB                             ;
                50   0C A7   B0  0093   313 50$:     MOVW    RK_ER(R7),R0            ; GET ERROR STATUS
                51   02 A7   32  0097   314          CVTWL   RK_WC(R7),R1            ; GET NEGATED COUNT REMAINING
                   51   02   C4  009B   315          MULL    #2,R1                   ; CONVERT TO BYTES
                55   18 A7   3C  009E   316          MOVZWL  RK_EC1(R7),R5           ; GET POSITION OF ERROR
                56   1A A7   B0  00A2   317          MOVW    RK_EC2(R7),R6           ; GET PATTERN
                     0008    31  00A6   318          BRW     ECC                     ; AND ATTEMPT ECC CORRECTION
                             00A9   319
                             00A9   320 ;
                             00A9   321 ; SUBROUTINE TO WAIT FOR CONTROLLER READY OR ERROR
                             00A9   322 ;
                             00A9   323
                             00A9   324 READY:                                    ;
              67   8080 8F   B3  00A9   325          BITW    #^X8080,(R7)            ;CONTROLLER READY OR ERROR?
                        F9   13  00AE   326          BEQL    READY                   ;IF EQL NO
                             05  00B0   327          RSB                             ;
```

```
                                00B1   329          .SBTTL  ECC - PERFORM ECC ERROR CORRECTION
                                00B1   330
                                00B1   331 ;++
                                00B1   332 ;
                                00B1   333 ; Functional description:
                                00B1   334 ;
                                00B1   335 ;       ATTEMPT ECC ERROR CORRECTION
                                00B1   336 ;
                                00B1   337 ; INPUTS:
                                00B1   338 ;
                                00B1   339 ;       R0      - RK_ER/RP_FR1  ERROR STATUS REGISTER
                                00B1   340 ;       R1      - NEGATIVE BYTE COUNT REMAINING
                                00B1   341 ;       R5      - ECC POSITION
                                00B1   342 ;       R6      - ECC PATTERN
                                00B1   343 ;       R8      - BYTE COUNT REMAINING AT START OF LAST TRANSFER
                                00B1   344 ;       R10     - starting address of transfer
                                00B1   345 ;       R11     - BLOCK NUMBER AT START OF TRANSFER
                                00B1   346 ;
                                00B1   347 ; Outputs:
                                00B1   348 ;
                                00B1   349 ;--
                                00B1   350
                                00B1   351 ECC:                                 ; ATTEMPT ECC CORRECTION
                                00B1   352
                                00B1   353 ;
                                00B1   354 ; Don't attempt an ECC correction if any of the following conditions apply:
                                00B1   355 ;
                                00B1   356 ;       the error was not a data check
                                00B1   357 ;       the error was a hard ECC error
                                00B1   358 ;       the transfer mode does not match the map-enabled position, i.e.,
                                00B1   359 ;               transfer is virtual, and mapping is not enabled, or v.v.
                                00B1   360 ;       no bytes have been transferred yet
                                00B1   361 ;
                                00B1   362
          58 50    0F    E1   00B1   363          BBC     #RK_ER_V_DCK,R0,RETRY   ; NOT DATA CHECK, RETRY
          57 50    06    E0   00B5   364          BBS     #RK_ER_V_ECH,R0,RETRY   ; HARD ECC ERROR, RETRY
             50    38    DB   00B9   365          MFPR    #PR$_MAPEN,R0           ; GET MAP ENABLE STATE
          14 AC    50    D1   00BC   366          CMPL    R0,MODE(AP)             ; SAME AS I/O MODE
                   4E    12   00C0   367          BNEQ    RETRY                   ; NO, CANT DO SIMPLE ECC
             51    58    C0   00C2   368          ADDL    R8,R1                   ; COMPUTE BYTES TRANSFERRED
                   49    13   00C5   369          BEQL    RETRY                   ; NONE, RETRY
                               00C7   370
                               00C7   371 ;
                               00C7   372 ; Appears to be a correctable data check. Attempt the correction.
                               00C7   373 ;
                               00C7   374
50   51    F7    8F    78   00C7   375          ASHL    #-9,R1,R0               ; CONVERT TO PAGE COUNT
             5B    50    C0   00CC   376          ADDL    R0,R11                  ; UPDATE BLOCK NUMBER
             5A    51    C0   00CF   377          ADDL    R1,R10                  ; UPDATE BYTE ADDRESS
             58    51    C2   00D2   378          SUBL    R1,R8                   ; DECREASE BYTES REMAINING
                   55    D7   00D5   379          DECL    R5                      ; MAKE POSITION 1 ORIGIN
   50      0200  C8    9E   00D7   380          MOVAB   512(R8),R0              ; REMAINING BYTES AT START OF BAD SECTOR
52   08    AC    50    C3   00DC   381          SUBL3   R0,SIZE(AP),R2          ; BYTES TRANSFERRED AT START OF ERROR SECTOR
             50    08    C4   00E1   382          MULL    #8,R0                   ; CONVERT BYTE COUNT TO BIT COUNT
             50    55    C2   00E4   383          SUBL    R5,R0                   ; COMPUTE CORRECTION FIELD WIDTH
                   19    15   00E7   384          BLEQ    20$                     ; BR IF NO CORRECTION NEEDED
             0D    50    D1   00E9   385          CMPL    R0,#RK_EC1_S_EPS        ; MINIMUM OF 13 AND BUFFER REMAINING
```

```
                          03   15  00EC   386          BLEQ    10$                         ; KEEP MINIMUM VALUE
                     50   0D   D0  00EE   387          MOVL    #RK_EC1_S_EPS,R0            ; LIMIT FIELD TO RK_EC1_S_EPS BITS
          51   04 BC42   50   55   EF  00F1   388 10$:   EXTZV   R5,R0,@BUF(AP)[R2],R1     ; GET FIELD TO BE CORRECTED
                     51   56   CC  00F8   389          XORL    R6,R1                       ; APPLY CORRECTION CODE
          04 BC42    50   55   51   F0  00FB   390          INSV    R1,R5,R0,@BUF(AP)[R2]       ; AND STORE IN BUFFER
                                     0102   391
                                     0102   392 ;
                                     0102   393 ; If the transfer is not complete, branch back to retry it.
                                     0102   394 ;
                                     0102   395
                          58   D5  0102     6 20$:   TSTL    R8                          ; CHECK FOR COUNT REMAINING
                          06   15  0104     7          BLEQ    30$                         ; NONE, EXIT
                     55   5B   D0  0106     8          MOVL    R11,R5                      ; GET WORKING COPY OF LBN
                        FEF4   31  0109   399          BRW     DM_DRIVER                   ; CONTINUE TRANSFER
                                     010C   400
                                     010C   401 ;
                                     010C   402 ; Transfer is complete. Return with success status code.
                                     010C   403 ;
                                     010C   404
                     50   01   3C  010C   405 30$:   MOVZWL  #SS$_NORMAL,R0              ; SET COMPLETION CODE
                                05  010F   406          RSB                             ; AND RETURN
                                     0110   407
                                     0110   408 ;
                                     0110   409 ; No ECC correction was possible.  Return and retry.
                                     0110   410 ;
                                     0110   411
                                     0110   412 RETRY:
               50   0054 8F   3C  0110   413          MOVZWL  #SS$_CTRLERR,R0             ; Set failure status
                                05  0115   414          RSB                             ; Return to BOOTDRIVR
                                     0116   415
58 45 2E 52 45 56 49 52 44 4D 44 00' 0116   416 DMNAME: .ASCIC   /DMDRIVER.EXE/           ; Driver filename
                                45  0122
                                0C  0116
                                     0123   417
                          00000123   0123   418 DM_DRVSIZ=.-DM_DRIVER
                                     0123   419
                                     0123   420          .END
```

```
$TABLE                          = 00000000 R  D  02
BTD$K_DM                        = 00000001    D
BUF                             = 00000004    D
DMNAME                            00000116 R  D  03
DM_DRIVER                         00000000 R  D  03
DM_DRVSIZ                       = 00000123    D
ECC                               000000B1 R  D  03
FUNC                            = 00000010    D
IO$_WRITELBLK                   = 00000020    D
LBN                             = 0000000C    D
MODE                            = 00000014    D
PR$_MAPEN                       = 00000038    D
READY                             000000A9 R  D  03
RETRY                             00000110 R  D  03
RK_AS                             0000000E    D
RK_BA                             00000004    D
RK_CS1                            00000000    D
RK_CS1_M_CDT                    = 00000400    D
RK_CS1_M_CERR                   = 00008000    D
RK_CS2                            00000008    D
RK_CS2_M_NED                    = 00001000    D
RK_CS2_M_SCLR                   = 00000020    D
RK_DA                             00000006    D
RK_DB                             00000014    D
RK_DC                             00000010    D
RK_DS                             0000000A    D
RK_DS_M_DDT                     = 00000100    D
RK_DS_M_DRDY                    = 00000080    D
RK_ECT                            00000018    D
RK_EC1_S_EPS                    = 0000000D    D
RK_EC2                            0000001A    D
RK_ER                             0000000C    D
RK_ER_V_DCK                     = 0000000F    D
RK_ER_V_ECH                     = 00000006    D
RK_MRT                            00000016    D
RK_MR2                            0000001C    D
RK_MR3                            0000001E    D
RK_SPR                            00000012    D
RK_WC                             00000002    D
RPB$W_UNIT                      = 00000064    D
SIZ...                          = 0000000B    D
SIZE                            = 00000008    D
SS$_CTRLERR                     = 00000054    D
SS$_NORMAL                      = 00000001    D
SS$_NOSUCHDEV                   = 00000908    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000020 (32.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000123 (291.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	181 (2)	181 (2)
BTD$K_DM	=00000001		181 (2)
BUF	=00000004	232 (3)	388 (4) 390 (4)
DMNAME	00000116-R	416 (4)	181 (2)
DM_DRIVER	00000000-R	238 (3)	181 (2) #-399 (4) 418 (4)
DM_DRVSIZ	=00000123	418 (4)	181 (2)
ECC	000000B1-R	351 (4)	#-318 (3)
FUNC	=00000010	235 (3)	#-246 (3)
IO$_WRITELBLK	=00000020		#-246 (3)
LBN	=0000000C	234 (3)	
MODE	=00000014	236 (3)	#-366 (4)
PR$_MAPEN	=00000038		#-365 (4)
READY	000000A9-R	324 (3)	#-257 (3) #-262 (3) #-279 (3) #-284 (3)
			#-301 (3) #-326 (3)
RETRY	00000110-R	412 (4)	#-363 (4) #-364 (4) #-367 (4) #-369 (4)
RK_BA	00000004		#-297 (3)
RK_CS1	00000000		#-261 (3) #-276 (3) #-278 (3) #-282 (3)
			#-300 (3) #-310 (3)
RK_CS1_M_CDT	=00000400		#-266 (3)
RK_CS1_M_CERR	=00008000		#-275 (3)
RK_CS2	00000008		#-256 (3) #-258 (3) #-269 (3) #-277 (3)
RK_CS2_M_NED	=00001000		#-269 (3)
RK_CS2_M_SCLR	=00000020		#-256 (3)
RK_DA	00000006		#-296 (3)
RK_DC	00000010		#-293 (3)
RK_DS	0000000A		#-264 (3) #-280 (3)
RK_DS_M_DDT	=00000100		#-264 (3)
RK_DS_M_DRDY	=00000080		#-280 (3)
RK_ECT	00000018		#-316 (3)
RK_EC1_S_EPS	=0000000D		#-385 (4) #-387 (4)
RK_EC2	0000001A		#-317 (3)
RK_ER	0000000C		#-313 (3)
RK_ER_V_DCK	=0000000F		#-363 (4)
RK_ER_V_ECH	=00000006		#-364 (4)
RK_WC	00000002		#-299 (3) #-314 (3)
RPB$W_UNIT	=00000064		#-255 (3)
SIZE	=00000008	233 (3)	#-381 (4)
SS$_CTRLERR	=00000054		#-413 (4)
SS$_NORMAL	=00000001		#-309 (3) #-405 (4)
SS$_NOSUCHDEV	=00000908		#-271 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	178 (2)	178 (2)
$BTDDEF	1	51 (2)	51 (2)
$DEFINI	1	51 (2)	51 (2) 52 (2) 53 (2) 54 (2) 55 (2) 56 (2) 57 (2) 70 (2)
$IODEF	17	52 (2)	52 (2)
$PRDEF	4	53 (2)	53 (2)
$RPBDEF	5	54 (2)	54 (2)
$SSDEF	21	55 (2)	55 (2)
$UBADEF	6	56 (2)	56 (2)
$UBIDEF	2	57 (2)	57 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.22
Command processing	141	00:00:00.74	00:00:01.55
Pass 1	538	00:00:13.08	00:00:15.84
Symbol table sort	0	00:00:01.87	00:00:02.10
Pass 2	128	00:00:02.26	00:00:02.47
Symbol table output	8	00:00:00.08	00:00:00.08
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	19	00:00:00.22	00:00:00.52
Assembler run totals	875	00:00:18.42	00:00:22.84

The working set limit was 1000 pages.
68417 bytes (134 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1256 non-local and 9 local symbols.
420 source lines were read in Pass 1, producing 0 object records in Pass 2.
45 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYS0.SYSMAINT]DS.MLB;218	1
DMA1:[SYS0.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	12

1219 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DMBTDRIVR/UPDA=(DMBTDRIVR.UPD,DMBTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S

Table of contents

(2) 63 DECLARATIONS
(3) 210 RB730:RB02/RB80 Bootstrap driver code

```
0000     1          .TITLE  DQBTDRIVR - RB730:RB02/RB80 BOOT DRIVER
0000     2          .IDENT  'V02-001'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1981                                                       *
0000     8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*****************************************************************************
0000    25 ;
0000    26
0000    27 ;++
0000    28 ; FACILITY:     BOOTS
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;       This module contains the bootstrap device driver for
0000    32 ;       the RB02 and RB80 disks on the RB730 controller.
0000    33 ;
0000    34 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    35 ;
0000    36 ; AUTHOR:  Greg Robert, CREATION DATE:  09-Jun-1981
0000    37 ;
0000    38 ; NOTE: The RB730 controller is supported by host microcode which is
0000    39 ;       activated each time a device register is accessed, and for
0000    40 ;       each block transferred.  Because of this, registers must be
0000    41 ;       handled in strict accordance with the RB730 software specification.
0000    42 ;       The following special registers protocols are significant:
0000    43 ;
0000    44 ;               DAR --  When the DAR is loaded some drive functions
0000    45 ;                       are initiated.  Consequently the function code
0000    46 ;                       must be loaded into the CSR (with CRDY set)
0000    47 ;                       prior to loading the DAR for seeks and transfers.
0000    48 ;                       For data transfers, the BAR and BCR must also
0000    49 ;                       be loaded prior to loading the DAR.
0000    50 ;
0000    51 ;               MPR --  The microcode keeps an internal disk address
0000    52 ;                       register for computing cylinder differance words
0000    53 ;                       for RB02 seek commands.  If this internal register
0000    54 ;                       falls out of sync with the actual disk address
0000    55 ;                       it can be reset by executing a "read header" command.
0000    56 ;                       Note that the header must actually be read (thru
0000    57 ;                       the MPR) in order to effect the reset.
```

```
                                0000     58 ;
                                0000     59 ; MODIFIED BY:
                                0000     60 ;
                                0000     61 ;--
```

```
                        0000     63          .SBTTL  DECLARATIONS
                        0000     64 ;
                        0000     65 ; INCLUDE FILES:
                        0000     66 ;
                        0000     .1          .library        /sys$library:lib/
                        0000     .2          .library        /$ds/
                        0000     .3          .library        /$diag/
                        0000     67
                        0000     68          $BTDDEF                         ; Boot device types
                        0000     69          $IODEF                          ; I/O function codes
                        0000     70          $PRDEF                          ; Processor Register definitions
                        0000     71          $RPBDEF                         ; RPB offsets
                        0000     72          $SSDEF                          ; Status codes
                        0000     73          $UBADEF                         ; UBA definitions
                        0000     74          $UBIDEF                         ; 11/750 UBA definitions
                        0000     75
                        0000     76 ;
                        0000     77 ; MACROS:
                        0000     78 ;
                        0000     79
                        0000     80 ;
                        0000     81 ; EQUATED SYMBOLS:
                        0000     82 ;
                        0000     83
                        0000     84 ;
                        0000     85 ; RB730:/RB02/RB80 REGISTER OFFSETS FROM CSR ADDRESS
                        0000     86 ;
                        0000     87          $DEFINI RB                      ; START OF REGISTER DEFINITIONS
                        0000     88
                        0000     89 $DEF     RB_CS           .BLKL   1       ;CONTROL STATUS REGISTER (CSR)
                        0004     90          _VIELD  RB_CS,0,<-              ;START OF CSR BIT DEFINITIONS
                        0004     91                  <DRDY,,M>,-             ; DRIVE READY
                        0004     92                  <FCODE,3>,-             ; FUNCTION CODE
                        0004     93                  <,2>,-                  ; RESERVED BITS
                        0004     94                  <IE,,M>,-               ; INTERRUPT ENABLE
                        0004     95                  <CRDY,,M>,-             ; CONTROLLER READY
                        0004     96                  <DS,2>,-                ; DRIVE SELECT
                        0004     97                  <OPI,,M>,-              ; OPERATION INCOMPLETE
                        0004     98                  <CRC,,M>,-              ; DATA CRC OR HEADER CRC OR DATA ECC
                        0004     99                  <DLT,,M>,-              ; DATA LATE OR HEADER NOT FOUND
                        0004    100                  <NXM,,M>,-              ; NON-EXISTENT MEMORY
                        0004    101                  <DE,,M>,-               ; DRIVE ERROR
                        0004    102                  <CE,,M>,-               ; COMPOSITE ERROR
                        0004    103                  <ATN,4>,-               ; DRIVE ATTENTION BITS
                        0004    104                  <ECC,2>,-               ; ECC STATUS
                        0004    105                  <SSEI,,M>,-             ; SKIP SECTOR ERROR INHIBIT
                        0004    106                  <SSE,,M>,-              ; SKIP SECTOR ERROR
                        0004    107                  <IR,,M>,-               ; IDC INTERRUPT REQUEST
                        0004    108                  <MTN,,M>,-              ; MAINTENANCE MODE
                        0004    109                  <TYP,,M>,-              ; DRIVE TYPE 1=RB80, 0=RB02
                        0004    110                  <ASSI,,M>,-             ; AUTOMATIC SKIP SECTOR INHIBIT
                        0004    111                  <TOI,,M>,-              ; TIME OUT INHIBIT (U-DIAG'S)
                        0004    112                  <FMT,,M>,-              ; RB0 FORMAT CONTROL
                        0004    113                  <,2>-                   ; RESERVED BITS
                        0004    114          >                               ;END CSR BIT DEFINITIONS
                        0004    115
                        0004    116 $DEF     RB_BA           .BLKL   1       ;BUS ADDRESS REGISTER (BAR)
```

```
                     0008   117
                     0008   118 $DEF     RB_BC           .BLKL   1       ;BYTE COUNT REGISTER (BCR)
                     000C   119
                     000C   120 $DEF     RB_DA           .BLKL   1       ;DISK ADDRESS REGISTER (DAR)
                     0010   121         _VIELD  RB_DA,0,<-               ;START OF DAR BIT DEFINITIONS
                     0010   122                 <SEC,8>,-                ; SECTOR
                     0010   123                 <TRK,8>,-                ; TRACK
                     0010   124                 <CYL,16>-                ; CYLINDER
                     0010   125         >                                ;END OF DAR BIT DEFINITIONS
                     0010   126
                     0010   127 $DEF     RB_MP           .BLKL   1       ;MULTIPURPOSE REGISTER (MPR)
                     0014   128         _VIELD  RB_MP,0,<-               ;RB02 STATUS WORD DEFINITIONS
                     0014   129                 <STA,3>,-                ; DRIVE STATE
                     0014   130                 <BH,,M>,-                ; BRUSH HOME
                     0014   131                 <HO,,M>,-                ; HEADS OUT
                     0014   132                 <CO,,M>,-                ; COVER OPEN
                     0014   133                 <HS,,M>,-                ; HEAD SELECT
                     0014   134                 <,1>,-                   ; RESERVED
                     0014   135                 <DSE,,M>,-               ; DRIVE SELECT ERROR
                     0014   136                 <VC,,M>,-                ; VOLUME CHECK
                     0014   137                 <WGE,,M>,-               ; WRITE GATE ERROR
                     0014   138                 <SPE,,M>,-               ; SPIN ERROR
                     0014   139                 <SKTO,,M>,-              ; SEEK TIME OUT
                     0014   140                 <WL,,M>,-                ; WRITE LOCK
                     0014   141                 <CHE,,M>,-               ; CURRENT HEAD ERROR
                     0014   142                 <WDE,,M>-                ; WRITE DATA ERROR
                     0014   143         >                                ;
                     0014   144         _VIELD  RB_MP,0,<-               ;GET STATUS COMMAND DEFINITIONS
                     0014   145                 <MRK,,M>,-               ; MARK (ALWAYS 1)
                     0014   146                 <STS,,M>,-               ; GET STATUS
                     0014   147                 <,1>,-                   ; RESERVED
                     0014   148                 <RST,,M>,-               ; RESET
                     0014   149         >
                     0014   150         _VIELD  RB_MP,0,<-               ;RB80 STATUS WORD DEFINITIONS
                     0014   151                 <SEC,5>,-                ; CURRENT RB80 SECTOR
                     0014   152                 <,3>,-                   ; RESERVED
                     0014   153                 <FLT,,M>,-               ; DRIVE FAULT
                     0014   154                 <PLGV,,M>,-              ; PLUG VALID
                     0014   155                 <SKE,,M>,-               ; SEEK ERROR
                     0014   156                 <ONCY,,M>,-              ; ON CYLINDER
                     0014   157                 <DRDY,,M>,-              ; DRIVE READY
                     0014   158                 <WTP,,M>,-               ; WRITE PROTECT
                     0014   159                 <,2>,-                   ; RESERVED
                     0014   160         >                                ;END MPR BIT DEFINITIONS
                     0014   161
                     0014   162 $DEF     RB_EC1          .BLKL   1       ;ECC PATTERN REGISTER (EPAR)
                     0018   163         _VIELD  RB_EC1,0,<-              ;START OF EC1 BIT DEFINITIONS
                     0018   164                 <POS,11>,-               ; STARTING BIT POSITION OF ECC ERROR
                     0018   165                 <,21>-                   ; RESERVED
                     0018   166         >                                ;END EC1 BIT DEFINITIONS
                     0018   167
                     0018   168 $DEF     RB_EC2          .BLKL   1       ;ECC POSITION REGISTER (EPOR)
                     001C   169         _VIELD  RB_EC2,0,<-              ;START OF EC2 BIT DEFINITIONS
                     001C   170                 <PAT,11>,-               ; PATTERN OF ECC ERROR BURST
                     001C   171                 <,21>-                   ; RESERVED
                     001C   172         >                                ;END EC2 BIT DEFINITIONS
                     001C   173
```

```
                        001C   174 $DEF    RB_CMD          .BLKL   1       ;AUXILLARY COMMAND REGISTER
                        0020   175         _VIELD  RB_CMD,0,<              ;START OF CMD BIT DEFINITIONS
                        0020   176                 <INIT,32>,-             ; SUBSYSTEM CLEAR <-- -1
                        0020   177         >                               ;END CMD BIT DEFINITIONS
                        0020   178
                        0020   179         $DEFEND RB                      ;END RB730:RB80/RB02 REGISTER DEFS
                        0000   180
                        0000   181 ;
                        0000   182 ; HARDWARE FUNCTION CODES
                        0000   183 ;
                        0000   184
                00000000 0000  185 F_NOP=0*2                               ;NO OPERATION
                00000006 0000  186 F_SEEK=3*2                              ;SEEK CYLINDER
                00000008 0000  187 F_READHEAD=4*2                          ;READ HEADER
                00000002 0000  188 F_WRITECHECK=1*2                        ;WRITE CHECK
                0000000A 0000  189 F_WRITEDATA=5*2                         ;WRITE DATA
                0000000C 0000  190 F_READDATA=6*2                          ;READ DATA
                00000004 0000  191 F_GETSTATUS=2*2                         ;GET STATUS
                00000[illegible] 0000  192 BTD$K_DQ = 3
                00000[illegible] 0000  193 PR$_SID_TYP730 = 3
                        0000   194
                        0000   195 ;
                        0000   196 ; OWN STORAGE:
                        0000   197 ;
                        0000   198
                        0000   199
                        0000   200 ;
                        0000   201 ; Boot driver table entry
                        0000   202 ;
                        0000   203
                        0000   204         $BOOT_DRIVER    CPUTYPE = PR$_SID_TYP730,- ; Must be VAX 11/730
                        0000   205                         DEVTYPE = BTD$K_DQ,-    ; Device type (DQ)
                        0000   206                         SIZE = DQ_DRVSIZ,-      ; Driver size
                        0000   207                         ADDR = DQ_DRIVER,-      ; Driver address
                        0000   208                         DRIVRNAME = DQNAME      ; Driver file name
```

```
                     0000   210          .SBTTL  RB730:RB02/RB80 Bootstrap driver code
                     0000   211
                     0000   212 ;++
                     0000   213 ;
                     0000   214 ; Inputs:
                     0000   215 ;
                     0000   216 ;       R3      - base address of adapter's register space
                     0000   217 ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                     0000   218 ;       R6      - contains 0
                     0000   219 ;       R7      - address of device's CSR
                     0000   220 ;       R8      - SIZE OF TRANSFER IN BYTES
                     0000   221 ;       R9      - address of the RPB
                     0000   222 ;       R10     - starting address of transfer (byte offset in first
                     0000   223 ;                 page ORed with starting map register number)
                     0000   224 ;
                     0000   225 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                     0000   226 ;       BUF(AP) - Buffer address
                     0000   227 ;       SIZE(AP)- Size of transfer
                     0000   228 ;
                     0000   229 ; Implicit inputs:
                     0000   230 ;
                     0000   231 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                     0000   232 ;
                     0000   233 ; Outputs:
                     0000   234 ;
                     0000   235 ;       R0 - status code
                     0000   236 ;
                     0000   237 ;               SS$_NORMAL      - successful transfer
                     0000   238 ;               SS$_CTRLERR     - fatal controller error
                     0000   239 ;
                     0000   240 ;       R3 - must be preserved
                     0000   241 ;
                     0000   242 ;       This routine destroys R1, R2, R4, R5, R6.
                     0000   243 ;
                     0000   244 ;
                     0000   245 ; Register useage during this routine:
                     0000   246 ;
                     0000   247 ;       R0 - scratch
                     0000   248 ;       R1 - byte count for current transfer / scratch
                     0000   249 ;       R2 - scratch
                     0000   250 ;       R3 - adaptor base address
                     0000   251 ;       R4 - unit number
                     0000   252 ;       R5 - logical block number for current transfer
                     0000   253 ;       R6 - disk address for current transfer / scratch
                     0000   254 ;       R7 - device CSR
                     0000   255 ;       R8 - remaining byte count
                     0000   256 ;--
                     0000   257
00000004             0000   258 BUF = 4
00000008             0000   259 SIZE = 8
00000010             0000   260 FUNC = 16
                     0000   261
                     0000   262 DQ_DRIVER:
                     0000   263
                     0000   264 ;
                     0000   265 ; COMPUTE ADDRESS OF DEVICE CSR
                     0000   266 ;
```

```
                57   0200 C3   DE  0000   267          MOVAL   ^X200(R3),R7            ; COMPUTE CORRECT DEVICE CSR
                                     0005   268
                                     0005   269 ;
                                     0005   270 ; POSITION AND STORE THE UNIT NUMBER IN R4 FOR GLOBAL USE
                                     0005   271 ;
                          54   D4    0005   272          CLRL    R4                      ; CLEAR R4
                  08    64   A9    F0  0007   273          INSV    RPB$W_UNIT(R9),#8,#2,R4 ; GET UNIT NUMBER
                        54   02          000B
                                     000D   274
                                     000D   275 ;
                                     000D   276 ; RESET DRIVE AND GET STATUS
                                     000D   277 ;
                  1C A7     01   CE  000D   278 10$:     MNEGL   #1,RB_CMD(R7)           ; INITIALIZE SUBSYSTEM
                                D0   0011   279          MOVL    #RB_MP_M_STS-           ; PUT GET STATUS
                                     0012   280                  !RB_MP_M_RST-           ; ... AND WITH RESET
                                     0012   281                  !RB_MP_M_MRK,-          ; ... AND MARK BIT
                  10 A7     0B         0012   282                  RB_MP(R7)               ; ... INTO DAR
                                54   C9  0015   283          BISL3   R4,-                ; MERGE UNIT NUMBER
                                     0017   284                  #F_GETSTATUS,-          ; ... AND FUNCTION
                            67   04          0017   285                  RB_CS(R7)               ; ... INTO CSR
                                00F9 30     0019   286          BSBW    READY                   ; WAIT FOR DRIVE AND CONTROLLER READY
                   50     10 A7   D0   001C   287          MOVL    RB_MP(R7),R0            ; FETCH STATUS WORD
                04000000 8F     D3   0020   288          BITL    #RB_CS_M_TYP,RB_CS(R7)  ; IS THIS AN RB80?
                                67          0026
                                15   12     0027   289          BNEQ    15$                     ; BRANCH IF SO
                                     0029   290
                                     0029   291 ;
                                     0029   292 ; CHECK RB02 STATUS
                                     0029   293 ;
                   50   05   00   ED  0029   294          CMPZV   #0,#5,R0,-              ; TEST BITS 04:00 OF STATUS FOR
                                1D          002D
                                     002E   295                  #RB_MP_M_HO-            ; ... HEADS OUT
                                     002E   296                  !RB_MP_M_BH-            ; ... BRUSHES HOME
                                     002E   297                  !5                      ; ... SEEK LINEAR MODE (READY TO GO)
                                DD   12     002E   298          BNEQ    10$                     ; LOOP IF NOT READY
                                     0030   299
                                     0030   300 ;
                                     0030   301 ; READ A HEADER TO MAKE SURE MICROCODE IS SYNCRONIZED WITH CURRENT
                                     0030   302 ; DISK POSITION
                                     0030   303 ;
                                54   C9  0030   304          BISL3   R4,-                ; MERGE UNIT NUMBER
                                     0032   305                  #F_READHEAD,-           ; ... AND FUNCTION
                            67   08          0032   306                  RB_CS(R7)               ; ... INTO CSR
                                00DE 30     0034   307          BSBW    READY                   ; WAIT FOR DRIVE AND CONTROLLER READY
                10 A7     10 A7   D1   0037   308          CMPL    RB_MP(R7),RB_MP(R7)     ; READ THE HEADER (UCODE DOESN'T LOOK
                                     003C   309                                          ; ...AT IT UNLESS WE ACCESS IT)
                                07   11     003C   310          BRB     20$                     ; CONTINUE IN COMMON
                                     003E   311
                                     003E   312
                                     003E   313 ;
                                     003E   314 ; CHECK RB80 STATUS
                                     003E   315 ;
                   50   05   08   ED  003E   316 15$:     CMPZV   #8,#5,R0,-              ; TEST BITS 08:12 OF STATUS FOR
                                1A          0042
                                     0043   317                  #<RB_MP_M_DRDY-         ; ...DRIVE READY
                                     0043   318                  !RB_MP_M_ONCY-          ; ...ON CYLINDER
                                     0043   319                  !RB_MP_M_PLGV>-         ; ...PLUG VALID
```

```
                                0043   320                      @-8                     ; ...SHIFT TO LOW BYTE
                          C8    12  0043   321 18$:    BNEQ     10$                     ; IF NOT, BRANCH TO WAIT FOR IT
                                0045   322
                                0045   323
                                0045   324 ;
                                0045   325 ; NOW CONVERT LOGICAL TO PHYSICAL -- THE COMPUTED DISK ADDRESS HAS THE
                                0045   326 ; FORM OF A LONG WORD WITH LOW BYTE=SECTOR, NEXT BYTE=TRACK, AND HIGH
                                0045   327 ; WORD = CYLINDER.
                                0045   328 ;
                                0045   329 ;              I)      CYLINDER = LBN / BLOCKS_PER_CYLINDER
                                0045   330 ;             II)      TRACK = REMAINDER(I) / BLOCKS_PER_TRACK
                                0045   331 ;            III)      RB02_SECTOR = REMAINDER(II) * 2
                                0045   332 ;             IV)      RB80_SECTOR = REMAINDER(II)
                                0045   333 ;
                                C045   334
                          51    D4  0045   335 20$:    CLRL     R1                      ; CLEAR HIGH PART OF DIVIDEND
                          56    D4  0C47   336         CLRL     R6                      ; CLEAR HIGH PART OF DIVIDEND
                04000000  8F    D3  0049   337         BITL     #RB_CS_M_TYP,RB_CS(R7)  ; IS THIS AN RB80?
                          67        004F
                          13    12  0050   338         BNEQ     25$                     ; BRANCH IF SO
                                0052   339
                                0052   340 ;
                                0052   341 ;COMPUTE RB02 DISK ADDRESS
                                0052   342 ;
                52    55    28    7B  0052   343         EDIV     #40/2*2,R5,R2,R0        ; R2 = DESIRED CYL, R0 = REMAINING BLKS
                            50        0056
                50    50    14    7B  0057   344         EDIV     #40/2,R0,R0,R6          ; R0 = DESIRED TRK, R6 = REMAINING BLKS
                            56        005B
                      56    02    C4  005C   345         MULL     #2,R6                   ; 2 SECTORS PER BLOCK
                51    28    56    C3  005F   346         SUBL3    R6,#40,R1               ; R1 = 256 BYTE SECTORS LEFT ON SURFACE
                            15    11  0063   347         BRB      27$                     ; CONTINUE IN COMMON
                                0065   348
                                0065   349 ;
                                0065   350 ;COMPUTE RB80 DISK ADDRESS INSTEAD
                                0065   351 ;
                 000001B2  8F    7B  0065   352 25$:    EDIV     #31*14,R5,R2,R0         ; R2 = DESIRED CYL, R0 = REMAINING BLKS
                50    52    55        0068
                50    50    1F    7B  006E   353         EDIV     #31,R0,R0,R6            ; R0 = DESIRED TRK, R6 = REMAINING BLKS
                            56        0072
                51    1F    56    C3  0073   354         SUBL3    R6,#31,R1               ; R1 = 512 BYTE SECTORS LEFT ON SURFACE
                      51    02    C4  0077   355         MULL     #2,R1                   ; R1 = 256 BYTE SECTORS LEFT ON SURFACE
                                007A   356
                                007A   357 ;
                                007A   358 ;IF FINAL TRANSFER USE REMAINING BYTE COUNT, ELSE USE REMAINDER OF TRACK
                                007A   359 ;
                  00000100  8F    C4  007A   360 27$:    MULL     #256,R1                 ; R1 = BYTES LEFT ON SURFACE
                            51        0080
                      51    58    D1  0081   361         CMPL     R8,R1                   ; ARE ADDITIONAL TRANSFERS REQUIRED?
                            03    1A  0084   362         BGTRU    28$                     ; BRANCH IF ANSWER YES
                      51    58    D0  0086   363         MOVL     R8,R1                   ; SET BYTE COUNT FOR FINAL TRANSFER
                                0089   364
                                0089   365 ;
                                0089   366 ;FORM FULL DISK ADDRESS (CYL, TRK, SEC)
                                0089   367 ;
                10    10    52    F0  0089   368 28$:    INSV     R2,#16,#16,R6           ; MOVE CYLINDER INTO HIGH WORD
                            56        008D
                08    08    50    F0  008E   369         INSV     R0,#8,#8,R6             ; MOVE TRACK INTO SECOND BYTE
```

```
                    56          0092
                                0093    370
                                0093    371 ;
                                0093    372 ; PERFORM SEEK -- NOTE: FOR RB730 SEEKS AND TRANSFERS, THE COMMAND
                                0093    373 ; MUST BE LOADED INTO THE CSR WITH CONTROLLER READY BIT SET, BEFORE
                                0093    374 ; WRITING TO THE DISK ADDRESS REGISTER !!!
                                0093    375 ;
                    54    C9    0093    376         BISL3   R4, -                       ; MERGE UNIT NUMBER
                                0095    377                 #F_SEEK-                    ; ... FUNCTION AND
                                0095    378                 !RB_CS_M_CRDY,-             ; ... SUPPRESS EXECUTION
          00000086  8F          0095    379                 RB_CS(R7)                   ; ... INTO CSR
                    67          009A
           0C A7    56    D0    009B    380         MOVL    R6,RB_DA(R7)                ; LOAD DISK ADDRESS IN DAR
          00000080  8F    CA    009F    381         BICL    #RB_CS_M_CRDY,RB_CS(R7)     ; INITIATE THE FUNCTION
                    67          00A5
                  006C  30    00A6    382         BSBW    READY                       ; WAIT FOR CONTROLLER AND DRIVE READY
                    51  19    00A9    383         BLSS    100$                        ; BRANCH IF ERROR
                                00AB    384
                                00AB    385
                                00AB    386 ;
                                00AB    387 ; SEEK IS COMPLETE -- EXECUTE TRANSFER FUNCTION
                                00AB    388 ;
      52    8C 8F     9A    00AB    389 60$:    MOVZBL  #F_READDATA-                ; ASSUME READ FUNCTION
                                00AF    390                 !RB_CS_M_CRDY,R2            ; ... WITH CONTROLLER READY
      20    10 AC     D1    00AF    391         CMPL    FUNC(AP),#IO$_WRITELBLK     ; IS IT A WRITE FUNCTION?
                        04  12    00B3    392         BNEQ    70$                         ; BRANCH IF NOT
      52    8A 8F     9A    00B5    393         MOVZBL  #F_WRITEDATA-               ; SET WRITE FUNCTION CODE
                                00B9    394                 !RB_CS_M_CRDY,R2            ; ... WITH CONTROLLER READY
                                00B9    395
                                00B9    396 ;
                                00B9    397 ; NOTE: THE DEVICE REGISTERS MUST BE LOADED IN THE PRESCRIBED ORDER!
                                00B9    398 ;       1) FUNCTION CODE LOADED INTO CSR WITH CRDY SET
                                00B9    399 ;       2) BYTE COUNT AND MEMORY ADDRESS (THESE TWO ARE REVERSABLE)
                                00B9    400 ;       3) DISK ADDRESS (FIRST HARDWARE SILO LOADED AT THIS TIME)
                                00B9    401 ;       4) CRDY CLEARED (SECOND SILO LOADED, XFER BEGINS)
                                00B9    402 ;
  67    52    54    C9    00B9    403 70$:    BISL3   R4,R2,-                     ; MERGE UNIT NUMBER AND FUNCTION
                                00BD    404                 RB_CS(R7)                   ;...INTO CSR
        08 A7     51    CE    00BD    405         MNEGL   R1,RB_BC(R7)                ; SET NEG TRANSFER BYTE COUNT
        04 A7     5A    D0    00C1    406         MOVL    R10,RB_BA(R7)               ; SET BUFFER ADDRESS
        0C A7     56    D0    00C5    407         MOVL    R6,RB_DA(R7)                ; SET DESIRED DISK ADDRESS
       00000080   8F    CA    00C9    408         BICL    #RB_CS_M_CRDY,RB_CS(R7)     ; INITIATE THE FUNCTION
                    67          00CF
                  0042  30    00D0    409         BSBW    READY                       ; WAIT FOR CONTROLLER AND DRIVE READY
                    27  19    00D3    410         BLSS    100$                        ; BRANCH ON ERROR
                                00D5    411
                                00D5    412 ;
                                00D5    413 ; UPDATE BUFFER ADDRESS, BLOCK NUMBER, AND BYTES REMAINING
                                00D5    414 ; FOR NEXT TRANSFER
                                00D5    415 ;
      5A    04 A7     D0    00D5    416         MOVL    RB_BA(R7),R10               ; UPDATE BUFFER ADDRESS
         58    51     C2    00D9    417         SUBL    R1,R8                       ; UPDATE BYTES LEFT TO TRANSFER
                  0D  13    00DC    418         BLEQU   90$                         ; BRANCH IF TRANSFER COMPLETE
       00000200   8F    C6    00DE    419         DIVL    #512,R1                     ; COMPUTE BLOCKS TRANSFERRED
                    51          00E4
         55    51     C0    00E5    420         ADDL    R1,R5                       ; UPDATE LOGICAL BLOCK NUMBER
                FF5A  31    00E8    421         BRW     20$                         ; CONTINUE TRANSFER
```

```
                                00EB   422
                                00EB   423 ;
                                00EB   424 ; TRANSFER COMPLETE - RETURN
                                00EB   425 ;
                                00EB   426
      51    08 AC    3C        00EB   427 90$:    MOVZWL  SIZE(AP),R1             ; SET TOTAL BYTES TRANSFERRED
                07   12        00EF   428         BNEQ    95$                     ; BRANCH IF ORIGINAL SIZE WAS TRANSFERRED
00008000    8F     D0        00F1   429         MOVL    #^X8000,R1              ; ELSE SIZE WAS FORCED TO 64K
                51             00F7
      50    01     3C        00F8   430 95$:    MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                05   00FB   431         RSB                             ; AND RETURN
                                00FC   432
                                00FC   433 ;
                                00FC   434 ; RETRY ERROR
                                00FC   435 ;
                                00FC   436
      58    08 AC    3C        00FC   437 100$:   MOVZWL  SIZE(AP),R8             ; RESTORE BYTE SIZE OF TRANSFER IN R8
                07   12        0100   438         BNEQ    110$                    ; BRANCH IF SIZE WAS LEGAL
00001F40    8F     D0        0102   439         MOVL    #8000,R8                ; ELSE FORCE TO 64K SIZE
                58             0108
      09    00     EF        0109   440 110$:   EXTZV   #0,#9,BUF(AP),R10       ; RESTORE BYTE OFFSET IN R10
   5A    04 AC               010C
   50    0054 8F    3C        010F   441         MOVZWL  #SS$_CTRLERR,R0          ; SET FATAL CONTROLLER ERROR
                05   0114   442         RSB                             ; AND ATTEMPT RETRY
                                0115   443
                                0115   444
                                0115   445 ;
                                0115   446 ; SUBROUTINE TO WAIT FOR CONTROLLER AND DRIVE READY OR ERROR
                                0115   447 ;
                                0115   448
                                0115   449 READY:
                D3   0115   450         BITL    #RB_CS_M_CRDY-          ; CONTROLLER READY
                                0116   451                 !RB_CS_M_CE,-           ; ... OR ERROR?
00008080    8F                  0116   452                 RB_CS(R7)               ; ... DEVICE CSR
                67             011B
                F7   13   011C   453         BEQL    READY                   ; LOOP UNTIL CRDY OR ERROR
                D3   011E   454         BITL    #RB_CS_M_DRDY-          ; DRIVE READY
                                011F   455                 !RB_CS_M_CE,-           ; ... OR ERROR?
00008001    8F                  011F   456                 RB_CS(R7)               ; ... DEVICE CSR
                67             0124
                EE   13   0125   457         BEQL    READY                   ; LOOP UNTIL DRDY OR ERROR
                05   0127   458         RSB                             ;
                                0128   459
49 52 44 51 44 00'  0128   460 DQNAME: .ASCIC  /DQDRIVER.EXE/          ; Driver file name
58 45 2E 52 45 56   012E
                45   0134
                0C   0128
                                0135   461
                00000135        0135   462 DQ_DRVSIZ=.-DQ_DRIVER
                                0135   463
                                0135   464         .END
```

```
$TABLE                          = 00000000 R  D  02
BTD$K_DQ                        = 00000003    D
BUF                             = 00000004    D
DQNAME                            00000128 R  D  03
DQ_DRIVER                         00000000 R  D  03
DQ_DRVSIZ                       = 00000135    D
FUNC                            = 00000010    D
F_GETSTATUS                     = 00000004    D
F_NOP                           = 00000000    D
F_READDATA                      = 0000000C    D
F_READHEAD                      = 00000008    D
F_SEEK                          = 00000006    D
F_WRITECHECK                    = 00000002    D
F_WRITEDATA                     = 0000000A    D
IO$_WRITELBLK                   = 00000020    D
PR$_SID_TYP730                  = 00000003    D
RB_BA                             00000004    D
RB_BC                             00000008    D
RB_CMD                            0000001C    D
RB_CS                             00000000    D
RB_CS_M_CE                      = 00008000    D
RB_CS_M_CRDY                    = 00000080    D
RB_CS_M_DRDY                    = 00000001    D
RB_CS_M_TYP                     = 04000000    D
RB_DA                             0000000C    D
RB_EC1                            00000014    D
RB_EC2                            00000018    D
RB_MP                             00000010    D
RB_MP_M_BH                      = 00000008    D
RB_MP_M_DRDY                    = 00001000    D
RB_MP_M_HO                      = 00000010    D
RB_MP_M_MRK                     = 00000001    D
RB_MP_M_ONCY                    = 00000800    D
RB_MP_M_PLGV                    = 00000200    D
RB_MP_M_RST                     = 00000008    D
RB_MP_M_STS                     = 00000002    D
READY                             00000115 R  D  03
RPB$W_UNIT                      = 00000064    D
SIZ...                          = 00000020    D
SIZE                            = 00000008    D
SS$_CTRLERR                     = 00000054    D
SS$_NORMAL                      = 00000001    D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000020 (32.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000135 (309.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	208 (2)	208 (2)
BTD$K_DQ	=00000003	192 (2)	208 (2)
BUF	=00000004	258 (3)	440 (3)
DQNAME	00000128-R	460 (3)	208 (2)
DQ_DRIVER	00000000-R	262 (3)	208 (2) 462 (3)
DQ_DRVSIZ	=00000135	462 (3)	208 (2)
FUNC	=00000010	260 (3)	#-391 (3)
F_GETSTATUS	=00000C04	191 (2)	#-284 (3)
F_NOP	=00000000	185 (2)	
F_READDATA	=0000000C	190 (2)	#-390 (3)
F_READHEAD	=00000008	187 (2)	#-305 (3)
F_SEEK	=00000006	186 (2)	#-378 (3)
F_WRITECHECK	=00000002	188 (2)	
F_WRITEDATA	=0000000A	189 (2)	#-394 (3)
IO$_WRITELBLK	=00000020		#-391 (3)
PR$_SID_TYP730	=00000003	193 (2)	208 (2)
RB_BA	00000004		#-406 (3) #-416 (3)
RB_BC	00000008		#-405 (3)
RB_CMD	0000001C		#-278 (3)
RB_CS	00000000		#-285 (3) #-288 (3) #-306 (3) #-337 (3)
			#-379 (3) #-381 (3) #-404 (3) #-408 (3)
			#-452 (3) #-456 (3)
RB_CS_M_CE	=00008000		#-451 (3) #-455 (3)
RB_CS_M_CRDY	=00000080		#-378 (3) #-381 (3) #-390 (3) #-394 (3)
			#-408 (3) #-451 (3)
RB_CS_M_DRDY	=00000001		#-455 (3)
RB_CS_M_TYP	=04000000		#-288 (3) #-337 (3)
RB_DA	0000000C		#-380 (3) #-407 (3)
RB_MP	00000010		#-282 (3) #-287 (3) #-308 (3)
RB_MP_M_BH	=00000008		#-297 (3)
RB_MP_M_DRDY	=00001000		#-318 (3)
RB_MP_M_HO	=00000010		#-296 (3)
RB_MP_M_MRK	=00000001		#-281 (3)
RB_MP_M_ONCY	=00000800		#-319 (3)
RB_MP_M_PLGV	=00000200		#-319 (3)
RB_MP_M_RST	=00000008		#-281 (3)
RB_MP_M_STS	=00000002		#-280 (3)
READY	00000115-R	449 (3)	#-286 (3) #-307 (3) #-382 (3) #-409 (3)
			#-453 (3) #-457 (3)
RPB$W_UNIT	=00000064		#-273 (3)
SIZE	=00000008	259 (3)	#-427 (3) #-437 (3)
SS$_CTRLERR	=00000054		#-441 (3)
SS$_NORMAL	=00000001		#-430 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	204 (2)	204 (2)
$BTDDEF	1	68 (2)	68 (2)
$DEFINI	1	68 (2)	68 (2) 69 (2) 70 (2) 71 (2) 72 (2) 73 (2) 74 (2) 87 (2)
$IODEF	17	69 (2)	69 (2)
$PRDEF	4	70 (2)	70 (2)
$RPBDEF	5	71 (2)	71 (2)
$SSDEF	21	72 (2)	72 (2)
$UBADEF	6	73 (2)	73 (2)
$UBIDEF	2	74 (2)	74 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.14	00:00:00.27
Command processing	143	00:00:00.77	00:00:01.63
Pass 1	909	00:00:13.60	00:00:18.47
Symbol table sort	0	00:00:01.91	00:00:02.09
Pass 2	167	00:00:02.39	00:00:02.81
Symbol table output	9	00:00:00.10	00:00:00.35
Psect synopsis output	6	00:00:00.02	00:00:00.02
Cross-reference output	30	00:00:00.20	00:00:00.23
Assembler run totals	1299	00:00:19.16	00:00:25.89

The working set limit was 1000 pages.
67105 bytes (132 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1228 non-local and 13 local symbols.
467 source lines were read in Pass 1, producing 0 object records in Pass 2.
45 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	0
DRB1:[DS.WORK]DS.MLB;218	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	12

1219 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DQBTDRIVR/UPDA=(DQBTDRIVR.UPD,DQBTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(2) 40 DECLARATIONS
(3) 107 DR$INT - DR32 INTERRUPT HANDLER

```
                        0000     1          .TITLE  DRINT - DR32 interrupt handler
                        0000     2
                        0000     3          .IDENT  /V01/
                    00000000     4          .PSECT  SEP, WRT, SHR, EXE, LONG
                        0000     5
                        0000     6 ;
                        0000     7 ;                                COPYRIGHT (c) 1978, 1980 BY
                        0000     8 ;                     DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        0000     9 ;
                        0000    10 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
                        0000    11 ; ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
                        0000    12 ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
                        0000    13 ; COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
                        0000    14 ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
                        0000    15 ; TRANSFERRED.
                        0000    16 ;
                        0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
                        0000    18 ; AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
                        0000    19 ; CORPORATIO .
                        0000    20 ;
                        0000    21 ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
                        0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        0000    23 ;
                        0000    24
                        0000    25 ;++
                        0000    26 ; FACILITY:     EXECUTIVE
                        0000    27 ;
                        0000    28 ; ABSTRACT:
                        0000    29 ;       THIS IS A DUMMY DR32 INTERRUPT HANDLER WHICH IS USED UNTIL
                        0000    30 ;       THE REAL DR32 DRIVER IS LOADED.
                        0000    31 ;
                        0000    32 ; ENVIRONMENT:  KERNEL MODE, NON-PAGED
                        0000    33 ;
                        0000    34 ; AUTHOR:  STEVE BECKHARDT,     CREATION DATE:  7-MAR-1979
                        0000    35 ;
                        0000    36 ; MODIFIED BY:
                        0000    37 ;
                        0000    38 ;--
```

```
                     0000     40          .SBTTL  DECLARATIONS
                     0000     41 ;
                     0000     42 ; INCLUDE FILES:
                     0000     43 ;
                     0000     44
                     0000     45 ;
                     0000     46 ; MACROS:
                     0000     47 ;
                     0000     48
                     0000     49          $IDBDEF                             ; IDB OFFSETS
                     0000     50
                     0000     51 ;
                     0000     52 ; EQUATED SYMBOLS:
                     0000     53 ;
                     0000     54
                     0000     55          ;
                     0000     56          ; DR32 DCR REGISTER DEFINITIONS
                     0000     57          ;
                     0000     58
                     0000     59          $DEFINI DR
                     0000     60 $DEF     DR_DCR          .BLKL   1           ; DR32 CONTROL REGISTER
                     0004     61          _VIELD  DR_DCR,0,<-
                     0004     62                  <ADPTYP,8>,-                ; ADAPTER TYPE
                     0004     63                  <ID2ERR,,M>,-               ; ID2 ERROR
                     0004     64                  <ID2TOS,2>,-                ; ID2 TIME-OUT STATUS
                     0004     65                  <,1>,-                      ; RESERVED
                     0004     66                  <ID1ERR,,M>,-               ; ID1 ERROR
                     0004     67                  <ID1TOS,2>,-                ; ID1 TIME-OUT STATUS
                     0004     68                  <RDS,,M>,-                  ; READ DATA SUBSTITUTE
                     0004     69                  <CRD,,M>,-                  ; CORRECTED READ DATA
                     0004     70                  <DCRHLT,,M>,-               ; DCR HALT
                     0004     71                  <DCRABT,,M>,-               ; DCR ABORT INTERRUPT
                     0004     72                  <PKTINT,,M>,-               ; PACKET INTERRUPT
                     0004     73                  <INTENB,,M>,-               ; INTERRUPT ENABLE
                     0004     74                  <,1>,-                      ; RESERVED
                     0004     75                  <PWR_UP,,M>,-               ; ADAPTER POWER UP
                     0004     76                  <PWR_DN,,M>,-               ; ADAPTER POWER DOWN
                     0004     77                  <EXTABT,,M>,-               ; EXTERNAL ABORT
                     0004     78                  <,1>,-                      ; RESERVED
                     0004     79                  <IMPDEP,6>,-                ; IMPLEMENTATION DEPENDENT BITS
                     0004     80          >
                     0004     81
                     0004     82 ; DCR CONTROL FIELD A CODES (USED WHEN WRITING TO DCR)
                     0004     83
00000100             0004     84 DCR_K_CLRPWRUP=^X100
00000200             0004     85 DCR_K_CLRPWRDN=^X200                         ; CLEAR POWER DOWN
00000300             0004     86 DCR_K_CLREXTABT=^X300                        ; CLEAR EXTERNAL ABORT
00000400             0004     87 DCR_K_CLRABTINT=^X400                        ; CLEAR ABORT INTERRUPT
00000500             0004     88 DCR_K_CLRINTENB=^X500                        ; CLEAR INTERRUPT ENABLE
00000600             0004     89 DCR_K_SETINTENB=^X600                        ; SET INTERRUPT ENABLE
00000700             0004     90 DCR_K_CLRHLT=^X700                           ; CLEAR HALT
                     0004     91
                     0004     92 ; DCR CONTROL FIELD B CODES (USED WHEN WRITING TO DCR)
                     0004     93
00001000             0004     94 DCR_K_CLRCRD=^X1000                          ; CLEAR CRD
00002000             0004     95 DCR_K_SETEXTABT=^X2000                       ; SET EXTERNAL ABORT
00003000             0004     96 DCR_K_CLRPKTINT=^X3000                       ; CLEAR PACKET INTERRUPT
```

```
00004000  0004    97 DCR_K_RESET=^X4000                      ; RESET
00005000  0004    98 DCR_K_SETOSQTST=^X5000                  ; SET OSEQ TEST
00006000  0004    99 DCR_K_CLROSQTST=^X6000                  ; CLEAR OSEQ TEST
          0004   100          $DEFEND DR
          0000   101
          0000   102 ;
          0000   103 ; OWN STORAGE:
          0000   104 ;
          0000   105
```

```
                                0000   107          .SBTTL  DR$INT - DR32 INTERRUPT HANDLER
                                0000   108 ;++
                                0000   109 ; FUNCTIONAL DESCRIPTION:
                                0000   110 ;
                                0000   111 ;       THIS MODULE IS A DUMMY DR32 INTERRUPT HANDLER WHICH IS USED
                                0000   112 ;       UNTIL THE REAL DR32 DRIVER IS LOADED.
                                0000   113 ;       THIS ROUTINE ALSO CONTAINS A DUMMY DR32 CONTROLLER INITIALIZATION
                                0000   114 ;       ENTRY POINT.
                                0000   115 ;
                                0000   116 ; CALLING SEQUENCE:
                                0000   117 ;
                                0000   118 ;       JSB FROM INTERRUPT VECTOR IN CRB
                                0000   119 ;
                                0000   120 ; INPUT PARAMETERS:
                                0000   121 ;
                                0000   122 ;       NONE
                                0000   123 ;
                                0000   124 ; IMPLICIT INPUTS:
                                0000   125 ;
                                0000   126 ;       THE STACK ON ENTRY IS AS FOLLOWS:
                                0000   127 ;
                                0000   128 ;                 0(SP)           ADDRESS OF IDB ADDRESS
                                0000   129 ;      4(SP) - 16(SP)           SAVED R2 - R5
                                0000   130 ;                20(SP)           INTERRUPT PC
                                0000   131 ;                24(SP)           INTERRUPT PSL
                                0000   132 ;
                                0000   133 ; OUTPUT PARAMETERS:
                                0000   134 ;
                                0000   135 ;       NONE
                                0000   136 ;
                                0000   137 ; IMPLICIT OUTPUTS:
                                0000   138 ;
                                0000   139 ;       NONE
                                0000   140 ;
                                0000   141 ; COMPLETION CODES:
                                0000   142 ;
                                0000   143 ;       NONE
                                0000   144 ;
                                0000   145 ; SIDE EFFECTS:
                                0000   146 ;
                                0000   147 ;       INTERRUPTS ARE DISABLED ON THE DR32
                                0000   148 ;
                                0000   149 ;--
                                0000   150
                                0000   151 DR$INT::
      53    9E    D0            0000   152          MOVL    @(SP)+,R3                       ; GET ADDRESS OF IDB
      54    63    D0            0003   153          MOVL    IDB$L_CSR(R3),R4                ; GET ADDRESS OF FIRST CSR
64    0100  8F    3C            0006   154          MOVZWL  #DCR_K_CLRPWRUP,DR_DCR(R4)      ; CLEAR POWER UP
64    0200  8F    3C            0008   155          MOVZWL  #DCR_K_CLRPWRDN,DR_DCR(R4)      ; CLEAR POWER DOWN
      52    8E    7D            0010   156          MOVQ    (SP)+,R2                        ; RESTORE REGISTERS
      54    8E    7D            0013   157          MOVQ    (SP)+,R4
                  02            0016   158          REI
                                0017   159
                                0017   160
                                0017   161 DR$INITIAL::                                    ; CONTROLLER INITIALIZATION
                                0017   162
                                0017   163
```

```
                64    4000 8F    3C  0017    164          MOVZWL  #DCR_K_RESET,(R4)              ; RESET DR (R4 POINTS TO FIRST CSR)
                      05         001C  001C    165          RSB
                                 001D  001D    166
                                 001D  001D    167
                                 001D  001D    168
                                 001D  001D    169          .END
```

```
DCR_K_CLRPWRDN = 00000200     D
DCR_K_CLRPWRUP = 00000100     D
DCR_K_RESET    = 00004000     D
DR$INITIAL       00000017 RG  D  01
DR$INT           00000000 RG  D  01
DR_DCR           00000000     D
IDB$B_TYPE       0000000A     D
IDB$C_LENGTH     00000034     D
IDB$K_LENGTH     00000034     D
IDB$L_ADP        00000010     D
IDB$L_CSR        00000000     D
IDB$L_OWNER      00000004     D
IDB$L_UCBLST     00000014     D
IDB$W_SIZE       00000008     D
IDB$W_UNITS      0000000C     D
SIZ...         = 00000006     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
SEP	0000001D (29.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	00000034 (52.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
DCR_K_CLRPWRDN	=00000200		#-155 (3)
DCR_K_CLRPWRUP	=00000100		#-154 (3)
DCR_K_RESET	=00004000		#-164 (3)
DR$INITIAL	00000017-R	161 (3)	
DR$INT	00000000-R	151 (3)	
DR_DCR	00000000		#-154 (3) #-155 (3)
IDB$L_CSR	00000000		#-153 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...	
$DEFINI	1	49 (2)	49 (2)	59 (2)
$IDBDEF	1	49 (2)	49 (2)	

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.22
Command processing	141	00:00:00.70	00:00:01.49
Pass 1	180	00:00:01.70	00:00:02.71
Symbol table sort	0	00:00:00.03	00:00:00.03
Pass 2	70	00:00:00.44	00:00:00.58
Symbol table output	1	00:00:00.02	00:00:00.02
Psect synopsis output	4	00:00:00.03	00:00:00.03
Cross-reference output	8	00:00:00.06	00:00:00.06
Assembler run totals	439	00:00:03.08	00:00:05.16

The working set limit was 1000 pages.
5883 bytes (12 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 57 non-local and 0 local symbols.
169 source lines were read in Pass 1, producing 0 object records in Pass 2.
14 pages of virtual memory were used to define 9 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	5

87 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DRINT/UPDA=(DRINT.UPD,DRINT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

```
;    0001  0      %title '*** DSLOAD DS$LOAD routine'
;    0002  0      MODULE DSLOAD (                                !Routines to handle DS$LOAD
;    0003  0                     IDENT = '07-09',
;    0004  0                     ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE)
;    0005  0                     ) =
;    0006  0      !
;    0007  0      ! COPYRIGHT (c) 1980, 1981,1984
;    0008  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0009  0      !
;    0010  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0011  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0012  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0013  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0014  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0015  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0016  0      ! REMAIN IN DEC.
;    0017  0      !
;    0018  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0019  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0020  0      ! CORPORATION.
;    0021  0      !
;    0022  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0023  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0024  0      !
;    0025  0      !++
;    0026  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0027  0      !
;    0028  0      ! ABSTRACT:
;    0029  0      !
;    0030  0      !       This module contains the routines necessary to handle
;    0031  0      !       the DS$LOAD routines.
;    0032  0      !
;    0033  0      ! AUTHOR:       Roger Riggs, CREATION DATE: 10-November-1979
;    0034  0      ! MODIFIED BY:
;    0035  0      !
;    0036  0      !               Dave Butenhof, 8-apr-1981, version 6.3
;    0037  0      !       01      In user mode, magtapes will only load one magtape block,
;    0038  0      !               irregardless of USZ field size, on $READ.  Therefore, loop
;    0039  0      !               on $READ until RMS$_EOF found, to make sure entire file is
;    0040  0      !               read.  Also, use $SPACE if VBN arg is specified, so it'll
;    0041  0      !               work on magtapes.
;    0042  0      !              -Dave Butenhof, 02-Jun-1981, version 6.3
;    0043  0      !       02      Change directory specs of libraries for flexibility
;    0044  0      !
;    0045  0      !       03      - Jack Stansbury, 28-Oct-1981, Version 6.5
;    0046  0      !               Changed PSECTs from SEP to WORK, CODE, and DATA.
;    0047  0      !
;    0048  0      !       04      Jack Stansbury, 10-Apr-1982, Version 6.7
;    0049  0      !               Changed the location of the following assignment statement:
;    0050  0      !                       addr = .address;
;    0051  0      !               in the routine, since if the return length is requested
;    0052  0      !               (ACTUALCOUNT() GEQ 5), the retlen is calculated to be the
;    0053  0      !               maximum of the actual loaded size (addr - address) or the
;    0054  0      !               XAB size. However, if the Length parameter is 0, the assignment
;    0055  0      !               statement above was not getting done, and the aboce calculation
;    0056  0      !               would have been wrong!
;    0057  0      !
```

```
;    0058  0     !       05      Richard Brown, 15-June-82,  Version 6.8
;    0059  0     !               Addition of code to differentiate between those diagnostics
;    0060  0     !               which have an image header and those which do not.  If the
;    0061  0     !               header is present, it is stored in a separate buffer area.
;    0062  0     !
;    0063  0     !       06      Richard Brown, 4-August-82, Version 6.9
;    0064  0     !               Change to allow loading files bigger than 64k.
;    0065  0     !
;    0066  0     !       07      Marion Baggett, 24-Sept-82,  Version 6.9
;    0067  0     !               Set addressing mode to fix trucation errors.
;    0068  0     !
;    0069  0     !       08      R.E.Muse, 15-Feb_84, version 6.14
;    0070  0     !               Added check of image header size for those diagnostics
;    0071  0     !               with new images headers created by VMS version 4.
;    0072  0     !
;    0073  0     !       09      Richard Brown, 25-Jun-84, Version 7.0
;    0074  0     !               Changed value of image header size checked for, since
;    0075  0     !               it changed between field test 1 and field test 2 of VMS V4.
;    0076  0     !--
;    0077  0
;    0078  1     BEGIN
;    0079  1     !
;    0080  1     ! INCLUDE FILES:
;    0081  1     !
;    0082  1
;    0083  1     LIBRARY '$DIAG';                !
;    0084  1
;    0085  1     LIBRARY '$DS';                  !
;    0086  1
;    0087  1     LIBRARY 'SYS$LIBRARY:LIB';
;    0088  1
;    0089  1     $DS_DSADEF
;    0090  1
;    0091  1     !
;    0092  1     ! TABLE OF CONTENTS:
;    0093  1     !
;    0094  1
;    0095  1     FORWARD ROUTINE
;    0096  1         DSX$LOAD;
;    0097  1
;    0098  1     !
;    0099  1     ! EXTERNAL REFERENCES:
;    0100  1     !
;    0101  1     !
;    0102  1     ! PSECT DEFINITIONS:
;    0103  1     !
;    0104  1
;    0105  1     PSECT                                                           !
;    0106  1         Plit    = Data (NoExecute, Share, NoWrite,                  !
;    0107  1                         Addressing_Mode (Long_Relative));           !
;    0108  1                                                                     !
;    0109  1     PSECT                                                           !
;    0110  1         Code    = Code (Execute, Share, NoWrite);                   !
;    0111  1                                                                     !
;    0112  1     PSECT                                                           !
;    0113  1         Own     = Work (NoExecute, NoShare, Write,                  !
;    0114  1                         Addressing_Mode (Long_Relative));           !
```

```
;    0115  1
;    0116  1      PSECT
;    0117  1          Global = Work (NoExecute, NoShare, Write,
;    0118  1                         Addressing_Mode (Long_Relative));
;    0119  1
;    0120  1      BIND                                                                    ![05]
;    0121  1              FMT_BAD_HDR =                                                   ![05]
;    0122  1       UPLIT (%ASCIC 'Routine DSX$LOAD has detected non-standard diag. or image header.!/Assuming no image header.!/');![0
;    0123  1
;    0124  1      EXTERNAL
;    0125  1              IMAGE_HEADER,                                                   ![05]
;    0126  1              IMAGE_HEADER_END;                                               ![05]
;    0127  1
```

```
;    0128  1     %SBTTL 'DSX$LOAD routine'
;    0129  1
;    0130  1     global routine dsx$load (file, default, length, address, retlen, retrec, vbn) =
;    0131  1
;    0132  1     !++
;    0133  1     ! Functional description:
;    0134  1     !
;    0135  1     !       This routine opens and read portions of the specified file using RMS
;    0136  1     !       into the buffer specified.
;    0137  1     !
;    0138  1     ! Formal parameters:
;    0139  1     !
;    0140  1     !       file    Address of a quadword descriptor of the file name to be used
;    0141  1     !       default Address of a quadword descriptor of the file name defaults
;    0142  1     !       length  Length of the area to receive the file in bytes.
;    0143  1     !       address Address of the buffer to receive the file
;    0144  1     !       retlen  Address of a longword to receive the length of the file (bytes)
;    0145  1     !       retrec  Address of a longword to receive record attributes of the file
;    0146  1     !       vbn     Virtual block of the file to start reading.
;    0147  1     !
;    0148  1     ! Side effects:
;    0149  1     !
;    0150  1     !       The current load media is used to locate the file
;    0151  1     !
;    0152  1     ! Completion codes:
;    0153  1     !
;    0154  1     !       Any RMS or system service return
;    0155  1     !--
;    0156  1
;    0157  2         begin
;    0158  2
;    0159  2         builtin
;    0160  2             actualcount;
;    0161  2
;    0162  2         label
;    0163  2             read;
;    0164  2
;    0165  2         local
;    0166  2             fab : $fab_decl,                        ! FAB for file access
;    0167  2             rab : $rab_decl,                        ! RAB for file access
;    0168  2             xab : $xabfhc_decl,                     ! XAB for file access
;    0169  2             addr,                                   ! address at end of load                    [01]
;    0170  2             status;                                 ! Status of RMS calls
;    0171  2
;    0172  2         map
;    0173  2             file : ref vector [2, long],            ! Length and address of file name
;    0174  2             default : ref vector [2, long];         ! Length and address of default file name
;    0175  2
;P   0176  2         $fab_init (fab = fab, fna = .file [1], fns = .file [0], dna = .default [1], dns = .default [0],
;    0177  2             xab = xab, fac = (get, bio));
;    0178  2         $xabfhc_init (xab = xab);
;    0179  2         $rab_init (rab = rab, fab = fab, rac = seq);
;    0180  2
;    0181  2         if not (status = $open (fab = fab)) then return .status;
;    0182  2
;    0183  2         addr = .address;                            ! Initial address to load                   [04]
;    0184  2
```

```
;    0185  2          if .length neq 0
;    0186  2          then
;    0187  2      read :
;    0188  3              begin
;    0189  3
;    0190  3              local                                   !                                                  [01]
;    0191  3                  first,                              !                                                  [01]
;    0192  3                  len;                                !                                                  [01]
;    0193  3
;    0194  3              if not (status = $connect (rab = rab)) then leave read;
;    0195  3
;    0196  3              if actualcount () geq 7                 ! If VBN specified                                 [01]
;    0197  3              then                                    !                                                  [01]
;    0198  3                  if .dsa$v_user                      ! if in user mode                                  [01]
;    0199  3                      and (.fab [fab$l_dev] and dev$m_sqd) neq 0      ! and magtape                      [01]
;    0200  3                  then                                !                                                  [01]
;    0201  4                      begin                           ! space forward                                    [01]
;    0202  4
;    0203  4                      local                           !                                                  [01]
;    0204  4                          blsize;                     ! size of magtape block                            [01]
;    0205  4
;    0206  4                      blsize = .fab [fab$w_bls]/512;  ! Size of block in pages                           [01]
;    0207  4                      if ((.vbn - 1)/.blsize)*.blsize neq .vbn - 1    ! If can't access                  [01]
;    0208  4                      then                            !                                                  [01]
;    0209  4                          return ds$_error;           ! return fatal error                               [01]
;    0210  4
;    0211  4                      rab [rab$l_bkt] = (.vbn - 1)/.blsize;   ! Number of tape blocks to skip           [01]
;    0212  4                      status = $space (rab = rab);    ! do it                                            [01]
;    0213  4
;    0214  4                      if not .status then leave read  !                                                  [01]
;    0215  4                      end                             !                                                  [01]
;    0216  3                  else                                !                                                  [01]
;    0217  3                      rab [rab$l_bkt] = .vbn;         !                                                  [01]
;    0218  3
;    0219  3              len = .length;                          ! Initial (max) length to load                     [01]
;    0220  3              first = 1;                              ! First time                                       [01]
;    0221  3
;    0222  3              while 1 do                              ! Loop on $READ until done                         [01]
;    0223  4                  begin                               !                                                  [01]
;    0224  4                  rab [rab$l_ubf] = .addr;
;    0225  4                  if .len gtr %X'FE00'                !                                                  [06]
;    0226  4                  then                                !                                                  [06]
;    0227  4                      rab [rab$w_usz] = %X'FE00'      !                                                  [06]
;    0228  4                  else                                !                                                  [06]
;    0229  4                      rab [rab$w_usz] = .len;
;    0230  4                  status = $read (rab = rab);
;    0231  4                  rab [rab$l_bkt] = 0;                ! Use NBP for future reads (if any)                [01]
;    0232  4                  addr = .addr + .rab [rab$w_rsz];    ! Set to read next section (or EOF)                [01]
;    0233  4
;    0234  4                  if not .status then exitloop;       ! Leave if no good                                 [01]
;    0235  4
;    0236  4                  len = .len - .rab [rab$w_rsz];      !                                                  [01]
;    0237  4
;    0238  4                  first = 0;                          ! not first loop anymore                           [01]
;    0239  4
;    0240  4                  if .len eql 0                       ! If no more room,                                 [01]
;    0241  4                  then                                !                                                  [01]
```

```
;    0242  5                    begin                          !                                                    [01]
;    0243  5                    status = rms$_normal;          !  that's OK                                         [01]
;    0244  5                    exitloop                       !                                                    [01]
;    0245  5                    end                            !                                                    [01]
;    0246  3                end;                               !                                                    [01]
;    0247  3
;    0248  3
;    0249  3            if not .first and .status eql rms$_eof ! If end of file, but on loop                        [01]
;    0250  3            then                                   !                                                    [01]
;    0251  3                status = rms$_normal;              !                                                    [01]
;    0252  3
;    0253  3            $disconnect (rab = rab);
;    0254  3
;    0255  3            ! If we are loading  something at address 200                                             ![05]
;    0256  3            ! -- i., e., a diagnostic program load, then see if there                                 ![05]
;    0257  3            ! is an image header.  (Do this by checking the first                                     ![05]
;    0258  3            ! longword loaded.) If the header exists, move                                            ![05]
;    0259  3            ! it to its own buffer space and move the rest of what                                    ![05]
;    0260  3            ! was read in down one page, so that the actual image will                                ![05]
;    0261  3            ! reside at 200.  If no header, the first record of the                                   ![05]
;    0262  3            ! image should be the "diagnostic header".  If the first                                  ![05]
;    0263  3            ! location of this header is not what the routine expects,                                ![05]
;    0264  3            ! the operator is warned of the discrepancy.                                              ![05]
;    0265  3            ! The reason for this is that if the image header or diagnostic                           ![05]
;    0266  3            ! header size is changed, this code will also have to be                                  ![05]
;    0267  3            ! changed, and the warning will be a reminder that the change                             ![05]
;    0268  3            ! must be made.  Note that if either of the header sizes                                  ![05]
;    0269  3            ! is changed, the routine will assume there is no image header.                           ![05]
;    0270  3
;    0271  3            IF .ADDRESS EQL %X'200'                                                                  ![05]
;    0272  3            THEN                                                                                     ![05]
;    0273  4               IF (..ADDRESS EQL %X'00300084')
;    0274  4               or (..ADDRESS EQL %X'003000A8')                                          ![08][09]
;    0275  3                    THEN                                                                            ![05]
;    0276  4                            BEGIN                                                                   ![05]
;    0277  4                            LOCAL PTR1, PTR2;                                                       ![05]
;    0278  4
;    0279  4                            ! Move the image header to its                                          ![05]
;    0280  4                            ! storage area for use by the                                           ![05]
;    0281  4                            ! debugger.                                                             ![05]
;    0282  4
;    0283  4                            PTR1 = IMAGE_HEADER;                                                    ![05]
;    0284  4                            PTR2 = .ADDRESS;                                                        ![05]
;    0285  4                            WHILE .PTR1 LSSU IMAGE_HEADER_END                                       ![05]
;    0286  4                            DO                                                                      ![05]
;    0287  5                                    BEGIN                                                           ![05]
;    0288  5                                    .PTR1 = ..PTR2;                                                 ![05]
;    0289  5                                    PTR1 = .PTR1 + 4;                                               ![05]
;    0290  5                                    PTR2 = .PTR2 + 4;                                               ![05]
;    0291  4                                    END;                                                            ![05]
;    0292  4
;    0293  4                            ! Now everything already loaded                                         ![05]
;    0294  4                            ! must be moved down one page,                                          ![05]
;    0295  4                            ! so that the diag. header                                              ![05]
;    0296  4                            ! starts at 200.                                                        ![05]
;    0297  4
;    0298  4                            PTR1 = .ADDRESS;                                                        ![05]
```

```
;    0299  4                          PTR2 = .ADDRESS + %X'200';                                  ![05]
;    0300  4                          WHILE .PTR2 LSSU .ADDR DO                                   ![05]
;    0301  5                                  BEGIN                                               ![05]
;    0302  5                                  .PTR1 = ..PTR2;                                     ![05]
;    0303  5                                  PTR1 = .PTR1 + 4;                                   ![05]
;    0304  5                                  PTR2 = .PTR2 + 4;                                   ![05]
;    0305  4                                  END;                                                ![05]
;    0306  4                          ADDR = .ADDR - %X'200';                                     ![05]
;    0307  4                          LEN = .LEN + %X'200';                                       ![05]
;    0308  4                          END                                                         ![05]
;    0309  3                  ELSE IF ..ADDRESS NEQ %X'00000058'
;    0310  3                   THEN
;    0311  3                          $DS_PRINTF (FMT_BAD_HDR);
;    0312  2              end;
;    0313  2
;    0314  2          if actualcount () geq 5 then .retlen = max (.xab [xab$l_ebk]^9 + .xab [xab$w_ffb] - 512,    !       [01]
;    0315  2              .addr - .address);                      ! Bigger of XAB size and actually loaded size       [01]
;    0316  2
;    0317  2          if actualcount () geq 6
;    0318  2          then
;    0319  3              begin
;    0320  3
;    0321  3              map
;    0322  3                  retrec : ref block [, byte];
;    0323  3
;    0324  3              retrec [0, 0, 16, 0] = .fab [fab$w_mrs];
;    0325  3              retrec [2, 0, 8, 0] = .fab [fab$b_rfm];
;    0326  3              retrec [3, 0, 8, 0] = .fab [fab$b_fsz];
;    0327  2              end;
;    0328  2
;    0329  2          $close (fab = fab);
;    0330  2          .status
;    0331  1      end;


                                                        .TITLE  DSLOAD *** DSLOAD DS$LOAD routine
                                                        .IDENT  \07-09\

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

4F 4C 24 58 53 44 20 65 6E 69 74 75 6F 52 5E 00000 P.AAA:  .ASCII  \^Routine DSX$LOAD has detected non-stand\  ;
64 65 74 63 65 74 65 64 20 73 61 68 20 44 41 0000F                                                          ;
                  64 6E 61 74 73 2D 6E 6F 6E 20 0001E                                                       ;
6D 69 20 72 6F 20 2E 67 61 69 64 20 64 72 61 00028        .ASCII  \ard diag. or image header.!/Assuming no \  ;
73 41 2F 21 2E 72 65 64 61 65 68 20 65 67 61 00037                                                          ;
               20 6F 6E 20 67 6E 69 6D 75 73 00046                                                          ;
2F 21 2E 72 65 64 61 65 68 20 65 67 61 6D 69 00050        .ASCII  \image header.!/\<0>                      ;
                                          00 0005F                                                          ;


                                        DSA$AL_APTMAIL=     65024
                                        DSA$GL_FLAGS=       65024
                                        DSA$GL_APTCOM=      65028
                                        DSA$GL_PASSES=      65032
                                        DSA$GL_UNITS=       65036
                                        DSA$GL_SECTNO=      65040
                                        DSA$GL_SID=         65044
                                        DSA$GL_MSGTYP=      65088
```

```
                                                                DSA$GL_ERRNO=       65092
                                                                DSA$GL_EVENT=       65096
                                                                DSA$GL_SUBTNO=      65100
                                                                DSA$GL_TESTNO=      65104
                                                                DSA$GL_PASSNO=      65108
                                                                DSA$GL_DEVLEN=      65112
                                                                DSA$GL_DEVNAM=      65116
                                                                DSA$GL_MSGPTR=      65128
                                                                FMT_BAD_HDR=        P.AAA
                                                                        .EXTRN  IMAGE_HEADER, IMAGE_HEADER_END
                                                                        .EXTRN  SYS$OPEN, SYS$CONNECT
                                                                        .EXTRN  SYS$SPACE, SYS$READ
                                                                        .EXTRN  SYS$DISCONNECT, DS$PRINTF
                                                                        .EXTRN  SYS$CLOSE

                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                        007C 00000      .ENTRY  DSX$LOAD, Save R2,R3,R4,R5,R6                   ; 0130
                                        5E      FF40    CE   9E 00002   MOVAB   -192(SP), SP
    0050    8F              00          6E              00   2C 00007   MOVC5   #0, (SP), #0, #80, $RMS_PTR                  ; 0177
                                                    70  AE       0000E
                                70      AE      5003    8F   B0 00010   MOVW    #20483  $RMS_PTR
                                C6      AD              22   90 00016   MOVB    #34, $RMS_PTR+22
                                CF      AD              02   90 0001A   MOVB    #2, $RMS_PTR+31
                                D4      AD              6E   9E 0001E   MOVAB   XAB, $RMS_PTR+36
                                        51      04      AC   D0 00022   MOVL    FILE, R1
                                DC      AD      04      A1   D0 00026   MOVL    4(R1), $RMS_PTR+44
                                        50      08      AC   D0 0002B   MOVL    DEFAULT, R0
                                E0      AD      04      A0   D0 0002F   MOVL    4(R0), $RMS_PTR+48
                                E4      AD              61   90 00034   MOVB    (R1), $RMS_PTR+52
                                E5      AD              60   90 00038   MOVB    (R0), $RMS_PTR+53
            2C              00          6E              00   2C 0003C   MOVC5   #0, (SP), #0, #44, $RMS_PTR                  ; 0178
                                        6E                   00041
                                        6E      2C1D    8F   B0 00042   MOVW    #11293, $RMS_PTR
    0044    8F              00          6E              00   2C 00047   MOVC5   #0, (SP), #0, #68, $RMS_PTR                  ; 0179
                                                2C  AE       0004E
                                2C      AE      4401    8F   B0 00050   MOVW    #17409, $RMS_PTR
                                        4A      AE      94   00056   CLRB    $RMS_PTR+30
                                68      AE      70      AE   9E 00059   MOVAB   FAB, $RMS_PTR+60
                                        70      AE      9F   0005E   PUSHAB  FAB                                             ; 0181
                            00000000G   00              01   FB 00061   CALLS   #1, SYS$OPEN
                                        56              50   D0 00068   MOVL    R0, STATUS
                                        03              56   E8 0006B   BLBS    STATUS, 1$
                                                0187    31   0006E   BRW     21$
                                        55      10      AC   D0 00071 1$:  MOVL    ADDRESS, R5                             ; 0183
                                        52              55   D0 00075   MOVL    R5, ADDR
                                                0C      AC   D5 00078   TSTL    LENGTH                                  ; 0185
                                                        5A   13 0007B   BEQL    3$
                                                0C      AE   9F 0007D   PUSHAB  RAB                                     ; 0194
                            00000000G   00              01   FB 00080   CALLS   #1, SYS$CONNECT
                                        56              50   D0 00087   MOVL    R0, STATUS
                                        4A              56   E9 0008A   BLBC    STATUS, 3$
                                        07              6C   91 0008D   CMPB    (AP), #7                                ; 0196
                                                        4D   1F 00090   BLSSU   5$
                        40 0000FE03     9F              04   E1 00092   BBC     #4, @#^X0000FE03, 4$                    ; 0198
                        3B      F0      AD              05   E1 0009A   BBC     #5, FAB+64, 4$                          ; 0199
                                        53      EC      AD   3C 0009F   MOVZWL  FAB+60, BLSIZE                           ; 0206
```

```
                        53 00000200   8F   C6 000A3          DIVL2   #512, BLSIZE                                    ;
                50      1C   AC       01   C3 000AA          SUBL3   #1, VBN, R0                                     ; 0207
                51           50       53   C7 000AF          DIVL3   BLSIZE, R0, R1                                  ;
                             53       51   C4 000B3          MULL2   R1, R3                                          ;
                             50       53   D1 000B6          CMPL    R3, R0                                          ;
                                      08   13 000B9          BEQL    2$                                              ;
                             50 00660002   8F   D0 000BB          MOVL    #6684674, R0                                ; 0209
                                      04   000C2              RET                                                     ;
                        64   AE           51   D0 000C3 2$:       MOVL    R1, RAB+56                                  ; 0211
                             2C   AE      9F   000C7              PUSHAB  RAB                                         ; 0212
                00000000G    00           01   FB 000CA          CALLS   #1, SYS$SPACE                               ;
                             56           50   D0 000D1          MOVL    R0, STATUS                                  ;
                             08           56   E8 000D4          BLBS    STATUS, 5$                                  ; 0214
                                  00DA    31   000D7 3$:         BRW     17$                                         ;
                        64   AE      1C   AC   D0 000DA 4$:       MOVL    VBN, RAB+56                                 ; 0217
                             54      0C   AC   D0 000DF 5$:       MOVL    LENGTH, LEN                                 ; 0219
                             53           01   D0 000E3          MOVL    #1, FIRST                                   ; 0220
                        50   AE           52   D0 000E6 6$:       MOVL    ADDR, RAB+36                                ; 0224
                0000FE00     8F           54   D1 000EA          CMPL    LEN, #65024                                 ; 0225
                                          08   15 000F1          BLEQ    7$                                          ;
                        4C   AE      FE00 8F   B0 000F3          MOVW    #-512, RAB+32                               ; 0227
                                          04   11 000F9          BRB     8$                                          ;
                        4C   AE           54   B0 000FB 7$:       MOVW    LEN, RAB+32                                 ; 0229
                             2C   AE      9F   000FF 8$:         PUSHAB  RAB                                         ; 0230
                00000000G    00           01   FB 00102          CALLS   #1, SYS$READ                                ;
                             56           50   D0 00109          MOVL    R0, STATUS                                  ;
                             64   AE      D4   0010C              CLRL    RAB+56                                      ; 0231
                        50   4E   AE      3C   0010F              MOVZWL  RAB+34, R0                                  ; 0232
                             52           50   C0 00113          ADDL2   R0, ADDR                                    ;
                             14           56   E9 00116          BLBC    STATUS, 9$                                  ; 0234
                        50   4E   AE      3C   00119              MOVZWL  RAB+34, R0                                  ; 0236
                             54           50   C2 0011D          SUBL2   R0, LEN                                     ;
                                          53   D4 00120          CLRL    FIRST                                       ; 0238
                                          54   D5 00122          TSTL    LEN                                         ; 0240
                                          C0   12 00124          BNEQ    6$                                          ;
                             56 00010001  8F   D0 00126          MOVL    #65537, STATUS                              ; 0243
                             10           53   E8 0012D 9$:       BLBS    FIRST, 10$                                  ; 0249
0001827A                     8F           56   D1 00130          CMPL    STATUS, #98938                              ;
                                          07   12 00137          BNEQ    10$                                         ;
                             56 00010001  8F   D0 00139          MOVL    #65537, STATUS                              ; 0251
                                  2C      AE   9F   00140 10$:       PUSHAB  RAB                                         ; 0253
00000000G                    00           01   FB 00143          CALLS   #1, SYS$DISCONNECT                          ;
00000200                     8F           55   D1 0014A          CMPL    R5, #512                                    ; 0271
                                          61   12 00151          BNEQ    17$                                         ;
00300084                     8F           65   D1 00153          CMPL    (R5), #3145860                              ; 0273
                                          09   13 0015A          BEQL    11$                                         ;
003000A8                     8F           65   D1 0015C          CMPL    (R5), #3145896                              ; 0274
                                          39   12 00163          BNEQ    16$                                         ;
                             51 00000000G EF   9E 00165 11$:       MOVAB   IMAGE_HEADER, PTR1                          ; 0283
                             50           55   D0 0016C          MOVL    R5, PTR2                                    ; 0284
                             53 00000000G EF   9E 0016F 12$:       MOVAB   IMAGE_HEADER_END, R3                        ; 0285
                             53           51   D1 00176          CMPL    PTR1, R3                                    ;
                                          05   1E 00179          BGEQU   13$                                         ;
                             81           80   D0 0017B          MOVL    (PTR2)+, (PTR1)+                            ; 0288
                                          EF   11 0017E          BRB     12$                                         ; 0285
                             51           55   D0 00180 13$:       MOVL    R5, PTR1                                    ; 0298
                             50      0200 C5   9E 00183          MOVAB   512(R5), PTR2                               ; 0299
```

```
                                52              50  D1 00188 14$:    CMPL    PTR2, ADDR                              ; 0300
                                                05  1E 0018B        BGEQU   15$                                     ;
                                81              80  D0 0018D        MOVL    (PTR2)+, (PTR1)+                        ; 0302
                                                F6  11 00190        BRB     14$                                     ; 0300
                                52      FE00    C2  9E 00192 15$:    MOVAB   -512(R2), ADDR                          ; 0306
                                54      0200    C4  9E 00197        MOVAB   512(R4), LEN                            ; 0307
                                                16  11 0019C        BRB     17$                                     ; 0273
                        00000058    8F          65  D1 0019E 16$:    CMPL    (R5), #88                               ; 0309
                                                0D  13 001A5        BEQL    17$                                     ;
                                00000000'       EF  9F 001A7        PUSHAB  FMT_BAD_HDR                             ; 0311
                        00000000G   9F          01  FB 001AD        CALLS   #1, @#DS$PRINTF                         ;
                                05              6C  91 001B4 17$:    CMPB    (AP), #5                                ; 0314
                                                1E  1F 001B7        BLSSU   19$                                     ;
                    50      10      AE          09  78 001B9        ASHL    #9, XAB+16, R0                          ;
                                51      14      AE  3C 001BE        MOVZWL  XAB+20, R1                              ;
                                50      FE00 C140   9E 001C2        MOVAB   -512(R1)[R0], R0                        ;
                                52              55  C2 001C8        SUBL2   R5, R2                                  ; 0315
                                52              50  D1 001CB        CMPL    R0, R2                                  ;
                                                03  18 001CE        BGEQ    18$                                     ;
                                50              52  D0 001D0        MOVL    R2, R0                                  ;
                            14  BC              50  D0 001D3 18$:    MOVL    R0, @RETLEN                             ; 0314
                                06              6C  91 001D7 19$:    CMPB    (AP), #6                                ; 0317
                                                12  1F 001DA        BLSSU   20$                                     ;
                                50      18      AC  D0 001DC        MOVL    RETREC, R0                              ; 0324
                                60      E6      AD  B0 001E0        MOVW    FAB+54, (R0)                            ;
                            02  A0      CF      AD  90 001E4        MOVB    FAB+31, 2(R0)                           ; 0325
                            03  A0      EF      AD  90 001E9        MOVB    FAB+63, 3(R0)                           ; 0326
                                                70  AE 9F 001EE 20$:    PUSHAB  FAB                                     ; 0329
                        00000000G   00          01  FB 001F1        CALLS   #1, SYS$CLOSE                           ;
                                50              56  D0 001F8 21$:    MOVL    STATUS, R0                              ; 0331
                                                04  001FB          RET                                             ;

; Routine Size:  508 bytes,    Routine Base:  CODE + 0000

;   0332  1
;   0333  1      END
;   0334  0      ELUDOM



;                                       PSECT SUMMARY
;
;       Name                        Bytes                           Attributes
;
;  DATA                                  96  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                                 508  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                                   Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
```

```
;       File                                    Total    Loaded  Percent      Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265                     784         4        0          85      00:00.2
;  DRB1:[DS.WORK]DS.L32;159                       653         0        0          42      00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7                18017        84        0         975      00:04.6
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE DSLOAD

; Size:          508 code + 96 data bytes
; Run Time:        00:16.7
; Elapsed Time:    00:29.9
; Lines/CPU Min:    1198
; Lexemes/CPU-Min: 21240
; Memory Used:  201 pages
; Compilation Complete
```

Table of contents

(1) 80 Libraries and Macros
(1) 86 The Dispatch Vectors

```
                                0000     1          .Title  DSVECS  *** DSVECS Module
                                0000    .1          .Ident  /01-09/
-1                              0000     3          .NoShow Conditionals
                                0000     4
                                0000     5 ;++
                                0000     6 ; COPYRIGHT (C) 1977,1980
                                0000     7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                                0000     8 ;
                                0000     9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                                0000    10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                                0000    11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                                0000    12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                0000    13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                                0000    14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                                C000    15 ; REMAIN IN DEC.
                                0000    16 ;
                                0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                0000    18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                0000    19 ; CORPORATION.
                                0000    20 ;
                                0000    21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                                0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                0000    23 ;--
```

```
                     0000     25 ;++
                     0000     26 ; Facility:
                     0000     27 ;
                     0000     28 ;       VAX Diagnostic Supervisor (part of the Resident-DS image).
                     0000     29 ;
                     0000     30 ; Abstract:
                     0000     31 ;
                     0000     32 ;       This module contains the dispatch vectors that the CRD-Loadable
                     0000     33 ;       routines need to access certain Resident-DS locations and routines.
                     0000     34 ;
                     0000     35 ; Environment:
                     0000     36 ;
                     0000     37 ;       Not applicable
                     0000     38 ;
                     0000     39 ; Author:
                     0000     40 ;
                     0000     41 ;       Jack Stansbury
                     0000     42 ;
                     0000     43 ; Date Written:
                     0000     44 ;
                     0000     45 ;       March 30, 1982
                     0000     46 ;
                     0000     47 ; Modifications:
                     0000     48 ;
                     0000     49 ;       01      Jack Stansbury, Version 1.0, June 22, 1982
                     0000     50 ;               Added two more vectors.
                     0000     51 ;
                     0000     52 ;       02      Jack Stansbury, Version 1.1, July 24, 1982
                     0000     53 ;               Added the CRD$GB_Menu_OnLine_Debug byte. If == 1, then
                     0000     54 ;               on-line CRD command is allowed. If == 0, the CRD command
                     0000     55 ;               is disallowed. It is == 0 by default.
                     0000     56 ;
                     0000     57 ;       03      Jack Stansbury, Version 1.1, July 27, 1982
                     0000     58 ;               Added the DS$GA_DS_Ctrl_C_First (formerly DS$GA_DSCntrlC) and
                     0000     59 ;               the DS$GA_DS_Ctrl_C_Second longwords to the dispatch list.
                     0000     60 ;
                     0000     61 ;       04      Jack Stansbury, Version 1.1, August 19, 1982
                     0000     62 ;               Added the Begin and DS$GL_CRD_Debug vectors. Also changed the
                     0000     63 ;               version number bytes from 1 to 2.
                     0000     64 ;
                     0000     65 ;       05      Jack Stansbury, Version 1.1, September 10, 1982
                     0000     66 ;               Added the Boot$L_R5 address. This is used by Menu Test to allow
                     0000     67 ;               it to determine how CRD was invoked.
                     0000     68 ;
                     0000     69 ;       06      Jack Stansbury, Version 1.1, September 12, 1982
                     0000     70 ;               Took out the one added in 05 above, and added the
                     0000     71 ;               DSR$Check_MenuTest_Set and the DSR$Check_AutoTest_Set routine
                     0000     72 ;               addresses.
                     0000     73 ;
                     0000     74 ;       07      Jack Stansbury, Version 1.1, October 26, 1982
                     0000     75 ;               Added the DS$GL_CRD_Flags longword. This is set up now as a
                     0000     76 ;               'flags' longword. It is now used for CRD_Debug (bit 1) and
                     0000     77 ;               for disabling ^C echoing (bit 0).
                     0000     .1 ;
                     0000     .2 ;       08      Jack Stansbury, Version ?.?, March 3, 1983
                     0000     .3 ;               Added a number of new (empty for now) dispatch vectors. These
                     0000     .4 ;               empty ones are to allow the debug vectors (in CRD) to be big
```

```
                        0000     .5 ;
                        0000     .6 ;
                        0000     .7 ;
                        0000     .8 ;        09      John Ciukaj     Version 6.11    4-April 1983
                        0000     .9 ;                - Changed name of CRD Online debug vector
                        0000    .10 ;                - Changed reference names to DSR$Check.. routines
                        0000    .11 ;
                        0000    78 ;--
```

enough to pass enough useful information back to the
DSR$Unload_CRD routine.

```
                        0000     80          .SubTitle       Libraries and Macros
                        0000     81
                        0000     82          .Library        /Sys$Library:Lib/       ; Use Lib last
                        0000     83          .Library        /$DS/                   ; Use DS.Mar second
                        0000     84          .Library        /$Diag/                 ; Use Diag.Mar first
```

```
                                0000    86          .SubTitle       The Dispatch Vectors
                           00000000     87          .Psect          Work, NoShr, NoExe, Wrt, Long
                                0000    88
                                0000    89 DS$GL_CRD_Flags::                                ; Analguous to the DS flags longword
                           00000000     0000    90          .Long   0                       ; all bits = 0 by default       [07]
                                0004    91
                                0004    .1          ;+
                                0004    .2          ; This longword is used for special debug of CRD by CRD Command
                                0004    .3          ;-
                                0004    .4
                                0004    .5 CRD$GL_CRD_Command_Debug::                       ;
                           00000000     0004    .6          .Long   0                       ; = 0 by default                [09]
-2                                      0008    94
                                0008    95          ;+
                                0008    96          ; These four bytes are for verification purposes. See the
                                0008    97          ; Exchange_Dispat_Vectors routine in the CRDLoad.B32 module for more
                                0008    98          ; of an explanation of these bytes.
                                0008    99          ;-
                                0008    100
                                0008    101 DS$AL_DS_Dispatch_Vectors::
                             19 0008    .1          .Byte   25                      ; The # of dispatch vectors     [08]
                             03 0009    .2          .Byte   3                       ; The positive version number   [08]
                             FD 000A    .3          .Byte   -3                      ; The negative version number   [08]
-3                           00 000B    105         .Byte   0                       ; The null byte
                                000C    106
                                000C    107         ;+
                                000C    108         ; These are the actual dispatch vectors. The first vector is the one that
                                000C    109         ; will contain the address of the routine in the CRD-Loadable image that
                                000C    110         ; will be called by the Print routine. Thus, the global label. The dispatch
                                000C    111         ; vectors now contain those routines and locations internal to the
                                000C    112         ; Resident-DS that the CRD routines will have to access. These vectors
                                000C    113         ; will be exchanged with another like-set of vectors that are part of the
                                000C    114         ; CRD-Loadable image.
                                000C    115         ;-
                                000C    116
                                000C    117 DS$GA_CRD_DS_Interface::                        ; Talk to the CRD image
                  00000000'     000C    118         .Address        DSV$Exit                ; To exit the Resident-DS
                                0010    119
                  00000000'     0010    120         .Address        DSX$Print               ; The DS Print routine
                                0014    121
                  00000000'     0014    122         .Address        DS$GL_Flags             ; The DS flags
                                0018    123
                  00000000'     0018    124         .Address        DS$GA_DS_Ctrl_C_First   ; The DS's 1st Control-C        [03]
                                001C    125                                                 ; . . . handler                 [03]
                                001C    126
                  00000000'     001C    127         .Address        DS$GA_PTable            ; The list of addresses of
                                0020    128                                                 ; ... PTables
                                0020    129
                  00000000'     0020    130         .Address        Exe$AloNonPaged         ; Allocate a block of Non-Paged
                                0024    131                                                 ; ... pool
                                0024    132
                                0024    133 DS$GA_User_Error_Text::                         ; The address of the user's
                  00000024'     0024    134         .Address        DS$GA_User_Error_Text   ; ... MsgAdr (from the ERRxxxx
                                0028    135                                                 ; ... macro) will be put here.
                                0028    136
                  00000000'     0028    137         .Address        Scan$Numeric            ; The address of the            [01]
                                002C    138                                                 ; ... Scan$Numeric routine in   [01]
```

```
                                0028   139                                     ; ... the PARSE module.          [01]
                                002C   140
                    00000000'   002C   141          .Address   Exe$DeaNonPaged ; The address of the Deallocate [01]
                                0030   142                                     ; ... Non-Paged pool routine.    [01]
                                0030   143
                    00000000'   0030   144          .Address   DS$GA_DS_Ctrl_C_Second ; The DS's 2nd Control-C  [03]
                                0034   145                                     ; . . . handler                  [03]
                                0034   146
                    00000000'   0034   147          .Address   Begin           ; The Begin label in KERNEL      [04]
                                0038   148
                                0038   149 DS$GL_CRD_Debug::                   ; Make it global to be in MAP    [04]
                    00000000    0038   150          .Long      0               ; If bit (1) = 1, CRD will print [07]
                                003C   151                                     ; . . . debug statements. If     [07]
                                003C   152                                     ; . . . bit (1) = 0, CRD won't   [07]
                                003C   153                                     ; . . . print them.              [07]
                                003C   154                                     ; Bit (0) is for disabling ^C    [07]
                                003C   155                                     ; . . . echoing.                 [07]
                                003C   156
                    00000000'   003C    .1          .Address   DSR$Check_MenuTest_Off_Set
                                0040    .2                                      ; Return a 1 iff
                                0040    .3                                      ; CRD MENU Offline is active     [09]
                                0040    .4
                    00000000'   0040    .5          .Address   DSR$Check_AutoTest_Off_Set
                                0044    .6                                      ; Return a 1 iff
                                0044    .7                                      ; CRD AUTO Offline is active     [09]
                                0044    .8
                                0044    .9          ;+
                                0044   .10          ; The above vectors are the ones that actually contain the VDS          [08]
                                0044   .11          ; addresses that CRD must know about. The ones below are unused except  [08]
                                0044   .12          ; by the Debug_Vector. Note: the total number of these vectors must     [08]
                                0044   .13          ; agree with the CRD$K_Total_Numb_Vectors constant in the CRD.R32       [08]
                                0044   .14          ; library module.                                                       [08]
                                0044   .15          ;-
                                0044   .16
                    00000070    0044   .17          .BlkL    <25-14>                ; CRD$K_Total_Numb_Vectors = 25 [08]
                                0070   .18                                         ; # of vectors above       = 14 [08]
-8                              0070   165          .End
```

```
BEGIN                           ********    X   01
CRD$GL_CRD_COMMAND_DEBUG        00000004 RG  D  01
DS$AL_DS_DISPATCH_VECTORS       00000008 RG  D  01
DS$GA_CRD_DS_INTERFACE          0000000C RG  D  01
DS$GA_DS_CTRL_C_FIRST           ********    X   01
DS$GA_DS_CTRL_C_SECOND          ********    X   01
DS$GA_PTABLE                    ********    X   01
DS$GA_USER_ERROR_TEXT           00000024 RG  D  01
DS$GL_CRD_DEBUG                 00000038 RG  D  01
DS$GL_CRD_FLAGS                 00000000 RG  D  01
DS$GL_FLAGS                     ********    X   01
DSR$CHECK_AUTOTEST_OFF_SET      ********    X   01
DSR$CHECK_MENUTEST_OFF_SET      ********    X   01
DSV$EXIT                        ********    X   01
DSX$PRINT                       ********    X   01
EXE$ALONONPAGED                 ********    X   01
EXE$DEANONPAGED                 ********    X   01
SCAN$NUMERIC                    ********    X   01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
WORK	00000070 (112.)	01 (1.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BEGIN	00000000-XR		147 (1)
CRD$GL_CRD_COMMAND_DEBUG	00000004-R	91.5 (1)	
DS$AL_DS_DISPATCH_VECTORS	00000008-R	101 (1)	
DS$GA_CRD_DS_INTERFACE	0000000C-R	117 (1)	
DS$GA_DS_CTRL_C_FIRST	00000000-XR		124 (1)
DS$GA_DS_CTRL_C_SECOND	00000000-XR		144 (1)
DS$GA_PTABLE	00000000-XR		127 (1)
DS$GA_USER_ERROR_TEXT	00000024-R	133 (1)	134 (1)
DS$GL_CRD_DEBUG	00000038-R	149 (1)	
DS$GL_CRD_FLAGS	00000000-R	89 (1)	
DS$GL_FLAGS	00000000-XR		122 (1)
DSR$CHECK_AUTOTEST_OFF_SET	00000000-XR		156.5 (1)
DSR$CHECK_MENUTEST_OFF_SET	00000000-XR		156.1 (1)
DSV$EXIT	00000000-XR		118 (1)
DSX$PRINT	00000000-XR		120 (1)
EXE$ALONONPAGED	00000000-XR		130 (1)
EXE$DEANONPAGED	00000000-XR		141 (1)
SCAN$NUMERIC	00000000-XR		137 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.10	00:00:00.23
Command processing	138	00:00:00.76	00:00:01.68
Pass 1	148	00:00:01.24	00:00:02.45
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	52	00:00:00.61	00:00:01.36
Symbol table output	0	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	3	00:00:00.06	00:00:00.08
Assembler run totals	380	00:00:02.83	00:00:05.85

The working set limit was 950 pages.
1780 bytes (4 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 18 non-local and 0 local symbols.
190 source lines were read in Pass 1, producing 0 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	0
DRB1:[DS.WORK]DS.MLB;218	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) DSVECS/UPDA=(DSVECS.UPD,DSVECS.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA

Table of contents

(1) 176 Libraries and Equated Symbols
(1) 194 Data Psect Declarations
(1) 206 DS Entry Points
(1) 1800 DS$SRVDISP_TABLE - Dispatch table for VDS services
(1) 1817 DSX$SERVICE_DISPATCHER - Service Dispatch Routine
(1) 1865 Resrvd_Entry Routine - For entry errors

```
0000     1          .TITLE  ENTRY   *** ENTRY DS service entry vectors
0000     2          .IDENT  /07-35/
0000     3          .NoShow Conditionals    ;                                                    [28]
0000     4
0000     5  ;++
0000     6  ; COPYRIGHT (c) 1977, 1982, 1983, 1984
0000     7  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     8  ;
0000     9  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    10  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    11  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    12  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    13  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    14  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    15  ; REMAIN IN DEC.
0000    16  ;
0000    17  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    18  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    19  ; CORPORATION.
0000    20  ;
0000    21  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    22  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    23
0000    24  ;++
0000    25  ; FACILITY:
0000    26  ;
0000    27  ;       VAX DIAGNOSTIC SUPERVISOR.
0000    28  ;
0000    29  ; ABSTRACT:
0000    30  ;
0000    31  ;       NONE
0000    32  ;
0000    33  ; ENVIRONMENT:
0000    34  ;
0000    35  ;       CEP AND SEP.
0000    36  ;
0000    37  ; AUTHOR:
0000    38  ;
0000    39  ;       KEN CHAPMAN     31-OCT-77       VERSION 01.
```

```
                        0000     41 ;
                        0000     42 ; MODIFIED BY:
                        0000     43 ;
                        0000     44 ;       01      CHANGE ENTRY VECTOR DISPATCH FROM 'BRW' TO 'JMP'
                        0000     45 ;
                        0000     46 ;
                        0000     47 ;       02      ADDED SYS$WAKE VECTOR.  REMOVED SYS$CRETVA VECTOR.
                        0000     48 ;
                        0000     49 ;
                        0000     50 ;       03      ADDED SYS$SETPRT ENTRY MASK.
                        0000     51 ;
                        0000     52 ;
                        0000     53 ;       04      CHANGED ENTRY MASKS FOR SYS$ASSIGN, SYS$DASSGN, SYS$CANCEL,
                        0000     54 ;               SYS$GTCHAN, SYS$ALLOC, SYS$DALLOC.
                        0000     55 ;
                        0000     56 ;
                        0000     57 ;       05      CHANGED ENTRY MASKS FOR SYS$EXPREG, SYS$CNTREG, DS$GETBUF,
                        0000     58 ;               AND DS$RELBUF.
                        0000     59 ;       06      ADDED NEW ENTRY NAMES: DS$SUMMARY, DS$CANWAIT, DS$ASKDATA,
                        0000     60 ;               DS$ASKVLD, DS$ASKLGCL, DS$ASKSTR, DS$SHOCHAN.
                        0000     61 ;       07      ADDED '+2' TO SKIP OVER LOCAL ENTRY MASKS IN SOME CASES.
                        0000     62 ;
                        0000     63 ;               Roger Riggs                 12-Jun-1978
                        0000     64 ;       08      Changed GETMEM, RELMEM, MOVVRT, MOVPHY to be reserved.
                        0000     65 ;               previously they were dummy routines in MEMMGT.
                        0000     66 ;               Also changed EXPREG, CNTREG, MMOFF, MMON, and SETPRT to
                        0000     67 ;               use .VECTOR pseudo op.
                        0000     68 ;       09      Changed names and entry vectors of:
                        0000     69 ;               RGETDATA        DSX$GETDATA
                        0000     70 ;               RGETADDRESS     DSX$GETADDRESS
                        0000     71 ;               RGETVIELD       DSX$GETVIELD
                        0000     72 ;               RGETLOGICAL     DSX$GETLOGICAL
                        0000     73 ;               RGETSTRING      DSX$GETSTRING
                        0000     74 ;               RGPHARD         DSX$GPHARD
                        0000     75 ;               RBREAK
                        0000     76 ;       10      Added DS$PRINTSIG entry point
                        0000     77 ;
                        0000     78 ;               Roger Riggs, 14-Jan-1979        Version 5.2
                        0000     79 ;       11      Activated DS$PROBE entry point at 101A0
                        0000     80 ;
                        0000     81 ;       12      David R. Butenhof  06-feb-80
                        0000     82 ;               Add entry point for EXE$SETAST system call
                        0000     83 ;
                        0000     84 ;               Roger Riggs, 24-MAR-1980
                        0000     85 ;       13      Added dummy entry points for SYS$LKWSET, SYS$SETPRI,
                        0000     86 ;               SYS$SETRWM, SYS$SETSWM, SYS$ULWSET
                        0000     87 ;
                        0000     88 ;               Roger Riggs, 27-May-1980, Version 5.4
                        0000     89 ;       14      Changed entry points for QIO routines to use .VECTOR
                        0000     90 ;               and JMP +2
                        0000     91 ;
                        0000     92 ;               Dave Butenhof   9-jun-1980, Version 5.4
                        0000     93 ;       15      Added entry vectors for VMS V2.0 driver support routines:
                        0000     94 ;               IOC$LOADUBAMAPA, IOC$GETBUTE, IOC$PUTBYTE, IOC$INITBUFWIND,
                        0000     95 ;               IOC$MOVFRUSER2, IOC$MOVFRUSER1, IOC$MOVTOUSER2, IOC$MOVTOUSER1.
                        0000     96 ;
                        0000     97 ;               Dave Butenhof   18-jun-1980,    version 5.5
```

```
0000     98 ;        16      Add driver entry vectors for SCH$LOCKW and SCH$UNLOCK, for
0000     99 ;                use by drivers doing buffered I/O.
0000    100 ;
0000    101 ;                Roger Riggs, 22-July-1980, Version 5.5
0000    102 ;        17      Added entry points for ???
0000    103 ;                Moved RMS$GL_ABSTIM, RMS$GQ_SYSTIME here from clock.
0000    104 ;
0000    105 ;                Roger Riggs, 27-Aug-1980
0000    106 ;        18      Added SYS$GET, SYS$READ, SYS$OPEN, SYS$CLOSE, SYS$CONNECT,
0000    107 ;                SYS$DISCONNECT for subset of RMS services.
```

```
0000   109 ;
0000   110 ;                    Dave Butenhof, 30-sep-1980, Version 6.1
0000   111 ;          19        Added entry point for DS$ATTACH--user-callable ATTACH
0000   112 ;                    command processing
0000   113 ;          20        Added entry vectors for COM$DRVDEALMEM, EXE$INSIOQ,
0000   114 ;                    IOC$PTETOPFN, and moved EXE$GQ_SYSTIME here from CLOCK to
0000   115 ;                    make it accessible to drivers.
0000   116 ;          21        Added entry point for DS$HELP--user-callable HELP command
0000   117 ;                    processing.
0000   118 ;          22        Add entry point for IOC$MOVFRUSER
0000   119 ;
0000   120 ;                    Dave Butenhof, 22-apr-1981, version 6.4
0000   121 ;          23        Add entry points for IOC$ALLOSPT, EXE$MAXACMODE, and
0000   122 ;                    EXE$READLOCK.  These were specified in release notes
0000   123 ;                    of version 6.3, but somehow the updates were lost.
0000   124 ;                    [also, define all libraries in source]
0000   125 ;
0000   126 ;          24        - dave butenhof, 11-Sep-1981, version 6.5
0000   127 ;                    Fix entries for event flag services.  They are all done
0000   128 ;                    as jumps directly to label: fix to allow for entry mask
0000   129 ;                    on routine.
0000   130 ;
0000   131 ;          25        - Jack Stansbury, 23-Oct-1981, Version 6.-
0000   132 ;                    Fixed a truncation error in the CODE Psect, and
0000   133 ;                    another possibly future one in the same Psect.
0000   134 ;
0000   135 ;          26        - Jack Stansbury, 23-Oct-1981, Version 6.-
0000   136 ;                    Added DS$BRANCH entry point, by putting it between
0000   137 ;                    DS$SUMMARY and DS$BGNSUB, and by taking out the blank
0000   138 ;                    one after DS$ASKSTR.
0000   139 ;
0000   140 ;          27        - Jack Stansbury, 16-Nov-1981, Version 6.-
0000   141 ;                    Changed the code at the DS$BREAK entry point. Put in a
0000   142 ;                    JMP to the label FAKE_JUMP, which is in the routine
0000   143 ;                    RESRVD_ENTRY. This was necessitated by the code at
0000   144 ;                    DS$BREAK occupying too many bytes for the entry point.
0000   145 ;
0000   146 ;          28        Marion Baggett, 12-Apr-1982,    Version 6.7
0000   147 ;                    Added entry point DSX$ERRPREP
0000   148 ;
0000   149 ;          29        Richard Brown, 27-May-82, Version 6.8
0000   150 ;                    Added entry point DS$SETPRIEXV.
0000   151 ;
0000   152 ;          30        Richard Brown, 24-June-82, Version 6.8
0000   153 ;                    Added entry point DS$MAPDBGBLOCK.
0000   154 ;
0000   155 ;          31        Richard Brown, 4-August-82, Version 6.9
0000   156 ;                    Added entry point DS$FREEDBGSYM.
0000   157 ;
0000   158 ;          32        Bob Bergazzi    23-Nov-82       Version 6.10
0000   159 ;                    Added entry point DS$LOADPCS.
0000   160 ;
0000   161 ;          33        Bob Bergazzi    May 16,1983     Version 6.11
0000   162 ;                    Changed the order of the .LIB statements.
0000   163 ;
0000   164 ;          34        Richard Brown   22-June-83      Version 6.12
0000   165 ;                    Added entry point DS$GETTERM.
```

```
0000   166 ;
0000   167 ;       35      Richard Brown   12-July-84      Version 7.0
0000   168 ;               Added entry point DS$SERVICE_DISPATCHER, to handle
0000   169 ;               all future additional services, since empty spots
0000   170 ;               in the dispatch vector area are in short supply.
0000   171 ;               Added DS$SRVDISP_TABLE, used by DS$SERVICE_DISPATCHER.
0000   172 ;               Added DSX$MOUNT and DSX$DISMOUNT to DS$SRVDISP_TABLE.
0000   173 ;               Changed DATA psect alignment from Byte to Quad.
0000   174 ;--
```

```
                        0000   176          .SBTTL  Libraries and Equated Symbols
                        0000   177 ;
                        0000   178 ; INCLUDE FILES:
                        0000   179 ;
                        0000   180          .LIBRARY        "SYS$LIBRARY:LIB"       ;                       [33]
                        0000   181          .LIBRARY        "$DS"                   ;                       [33]
                        00     182          .LIBRARY        "$DIAG"                 ;                       [33]
                        000U   183
                        0000   184 ;
                        0000   185 ; MACROS:
                        0000   186 ;
                        0000   187
                        0000   188 ;
                        0000   189 ; EQUATED SYMBULS:
                        C000   190 ;
0000000D                0000   191          CR      = 13
0000000A                0000   192          LF      = 10
```

```
                                    0000   194          .SUBTITLE       Data Psect Declarations
                                    0000   195
                                    0000   196 ;
                                    0000   197 ; OWN STORAGE:
                                    0000   198 ;
                                00000000   199          .PSECT  DATA, SHR, NOEXE, NOWRT, quad                    ; [35]
                                    0000   200
                                    0000   201          MODNAM  ENTRY
                                    0006   202
                                    0006   203 T_RESRVD:
56 52 45 53 45 52 20 3F 3F 0A 0D 00' 0006   204          .ASCIC  <CR><LF>\?? RESERVED DIAGNOSTIC/SYSTEM SERVICE CALL.\
49 54 53 4F 4E 47 41 49 44 20 44 45  0012
52 45 53 20 4D 45 54 53 59 53 2F 43  001E
       2E 4C 4C 41 43 20 45 43 49 56  002A
                                 2D  C006
```

```
0034    206          .SBTTL  DS Entry Points
0034    207 ;++
0034    208 ; FUNCTIONAL DESCRIPTION:
0034    209 ;
0034    210 ;       THIS SECTION PROVIDES A WELL DEFINED LINK B       THE DIAGNOSTIC
0034    211 ;       SUPERVISOR AND THE DIAGNOSTIC PROGRAM.  EAC       INE STARTS ON A
0034    212 ;       FIXED QUADWORD BOUNDRY WHICH CONTAINS THE A       RIATE LINK TO
0034    213 ;       THE BODY OF THE ROUTINE.  THERE ARE FOUR SETS OF LINK VECTORS,
0034    214 ;       ONE FOR EACH TYPE OF PROGRAM:  CEP FUNCTIONAL, CEP REPAIR,
0034    215 ;       SEP FUNCTIONAL, SEP REPAIR.
0034    216 ;
0034    217 ; CALLING SEQUENCE:
0034    218 ;
0034    219 ;       ALL ROUTINES ARE CALLED VIA THE STANDARD VAX CALL PROCEDURE,
C034    220 ;       I.E., CALLG AND CALLS INSTRUCTIONS.
0034    221 ;
0034    222 ; INPUT PARAMETERS:
0034    223 ;
0034    224 ;       SEE SPECIFIC ROUTINE.
0034    225 ;
0034    226 ; IMPLICIT INPUTS:
0034    227 ;
0034    228 ;       SEE SPECIFIC ROUTINE.
0034    229 ;
0034    230 ; OUTPUT PARAMETERS:
0034    231 ;
0034    232 ;       SEE SPECIFIC ROUTINE.
0034    233 ;
0034    234 ; IMPLICIT OUTPUTS:
0034    235 ;
0034    236 ;       SEE SPECIFIC ROUTINE.
0034    237 ;
0034    238 ; COMPLETION CODES:
0034    239 ;
0034    240 ;       SEE SPECIFIC ROUTINE.
0034    241 ;
0034    242 ; SIDE EFFECTS:
0034    243 ;
0034    244 ;       SEE SPECIFIC ROUTINE.
0034    245 ;
0034    246 ;--
```

```
                    00000000   248          .PSECT  $BASE, SHR, NOEXE, NOWRT, QUAD
                        0000   249
                        0000   250 DS$ENTRY:                                  ; Address 10000 (X)
                01E9     31 0000   251          BRW     LEAP_BOOT              ; STAND ALONE ENTRY POINT.
                        0003   252
                        0003   253          .ALIGN  LONG
                        0004   254 DS$APT_ENTRY:
                01ED     31 0004   255          BRW     LEAP_APT               ; APT ENTRY POINT
                        0007   256
                        0007   257          .ALIGN  QUAD
                        0008   258 ; RESERVED ENTRY POINT
                    0000 0008   259          .WORD   ^M<>                   ; ENTRY MASK.
            0000001D'EF  17 000A   260          JMP     RESRVD_ENTRY
```

```
                     0010   262 ;+
                     0010   263 ; 4.1.11         INITIALIZE CODE SERVICES.
                     0010   264 ;-
                     0010   265         .ALIGN  QUAD
                     0010   266 DS$ENDPASS::                          ; END OF PASS ENTRY POINT.
                0000' 0010   267         .VECTOR DSX$ENDPASS
    00000002'EF   17  0012   268         JMP     DSX$ENDPASS+2        ;
                     0018   269
                     0018   270         .ALIGN  QUAD
                     0018   271 DS$GPHARD::                           ; GET POINTER TO HARDWARE PARAMETER TABLE.
                0000' 0018   272         .VECTOR DSX$GPHARD
    00000002'EF   17  001A   273         JMP     DSX$GPHARD+2
                     0020   274
                     0020   275 ;+
                     0020   276 ; 4.1.12         CLEANUP CODE CALL.
                     0020   277 ;-
                     0020   278         .ALIGN  QUAD
                     0020   279 DS$ABORT::                            ; ABORT PROGRAM.
                0000' 0020   280         .VECTOR DSX$ABORT
    00000002'EF   17  0022   281         JMP     DSX$ABORT+2
                     0028   282
                     0028   283 ;+
                     0028   284 ; 4.1.13         SUMMARY REPORT CALL.
                     0028   285 ;-
                     0028   286         .ALIGN  QUAD
                     0028   287 DS$SUMMARY::
                     0028   288 DS$DOSUMMARY::                        ; EXECUTE SUMMARY REPORT ROUTINE.
                0000' 0028   289         .VECTOR DSX$DOSUMMARY
    00000002'EF   17  002A   290         JMP     DSX$DOSUMMARY+2
```

```
                        0030   292 ;+
                        0030   293 ; 4.2.1          PROGRAM CONTROL SERVICES.
                        0030   294 ;-
                        0030   295         .ALIGN  QUAD
                        0030   296 DS$BGNSUB::                                ; BEGIN SUBTEST ENTRY POINT.
                0000'   0030   297         .VECTOR DSX$BGNSUB
    00000002'EF   17    0032   298         JMP     DSX$BGNSUB+2
                        0038   299
                        0038   300         .ALIGN  QUAD
                        0038   301 DS$ENDSUB::                                ; END SUBTEST ENTRY POINT.
                0000'   0038   302         .VECTOR DSX$ENDSUB
    00000002'EF   17    003A   303         JMP     DSX$ENDSUB+2
                        0040   304
                        0040   305         .ALIGN  QUAD
                        C040   306 DS$CKLOOP::                                ; CHECK LOOP ENRTY POINT.
                0000'   0040   307         .VECTOR DSX$CKLOOP
    00000002'EF   17    0042   308         JMP     DSX$CKLOOP+2
                        0048   309
                        0048   310         .ALIGN  QUAD
                        0048   311 DS$INLOOP::                                ; IN LOOP ENTRY POINT.
                0000'   0048   312         .VECTOR DSX$INLOOP
    00000002'EF   17    004A   313         JMP     DSX$INLOOP+2
                        0050   314
                        0050   315         .ALIGN  QUAD
                        0050   316 DS$ESCAPE::                                ; ESCAPE ENTRY POINT.
                0000'   0050   317         .VECTOR DSX$ESCAPE
    00000002'EF   17    0052   318         JMP     DSX$ESCAPE+2
                        0058   319
                        0058   320         .ALIGN  QUAD
                        0058   321 DS$BREAK::                                 ; BREAK FOR DYNAMIC SERVICES.
                  0000  0058   322         .WORD   ^M<>                       ; ENTRY MASK.
    00000032'EF   17    005A   323         JMP     FAKE_JUMP                  ; Go to Fake_Jump, which JSB's to KB_CHECK,
                        0060   324                                            ; and then RETurns.
                        0060   325
                        0060   326         .ALIGN  QUAD
                        0060   327 DS$WAITMS::                                ; WAIT MILLISECONDS ENTRY POINT.
                0000'   0060   328         .VECTOR DSX$WAITMS
    00000002'EF   17    0062   329         JMP     DSX$WAITMS+2
                        0068   330
                        0068   331         .ALIGN  QUAD
                        0068   332 DS$WAITUS::                                ; WAIT MICROSECONDS ENTRY POINT.
                0000'   0068   333         .VECTOR DSX$WAITUS
    00000002'EF   17    006A   334         JMP     DSX$WAITUS+2
                        0070   335
                        0070   336         .ALIGN  QUAD
                        0070   337 DS$CANWAIT::
                        0070   338 DS$ABORTWAIT::                             ; CANCEL WAIT ENTRY POINT.
                0000'   0070   339         .VECTOR DSX$CANWAIT
    00000002'EF   17    0072   340         JMP     DSX$CANWAIT+2
                        0078   341
                        0078   342         .ALIGN  QUAD
                        0078   343 DS$CNTRLC::
                0000'   0078   344         .VECTOR DSX$CNTRLC
    00000002'EF   17    007A   345         JMP     DSX$CNTRLC+2
```

```
                        0080   347 ;+
                        0080   348 ; 4.2.2          MESSAGE HANDLER SERVICES.
                        0080   349 ;-
                        0080   350          .ALIGN  QUAD
                        0080   351 DS$ASKDATA::
                        0080   352 DS$GETDATA::                              ; GET DATA ENTRY POINT.
                0000'   0080   353          .VECTOR DSX$GETDATA
    00000002'EF   17    0082   354          JMP     DSX$GETDATA+2
                        0088   355
                        0088   356          .ALIGN  QUAD
                        0088   357 DS$ASKVLD::
                        0088   358 DS$GETVIELD::                             ; GET A VIELD ENTRY POINT.
                0000'   0088   359          .VECTOR DSX$GETVIELD
    00000002'EF   17    008A   360          JMP     DSX$GETVIELD+2
                        0090   361
                        0090   362          .ALIGN  QUAD
                        0090   363 DS$ASKADR::
                        0090   364 DS$GETADDRESS::                           ; GET AN ADDRESS ENTRY POINT.
                0000'   0090   365          .VECTOR DSX$GETADDRESS
    00000002'EF   17    0092   366          JMP     DSX$GETADDRESS+2
                        0098   367
                        0098   368          .ALIGN  QUAD
                        0098   369 DS$ASKLGCL::
                        0098   370 DS$GETLOGICAL::                           ; GET A LOGICAL DECISION ENTRY POINT.
                0000'   0098   371          .VECTOR DSX$GETLOGICAL
    00000002'EF   17    009A   372          JMP     DSX$GETLOGICAL+2
                        00A0   373
                        00A0   374          .ALIGN  QUAD
                        00A0   375 DS$ASKSTR::
                        00A0   376 DS$GETSTRING::                           ; GET AN ASCII STRING ENTRY POINT.
                0000'   00A0   377          .VECTOR DSX$GETSTRING
    00000002'EF   17    00A2   378          JMP     DSX$GETSTRING+2
                        00A8   379
                        00A8   380
                        00A8   381          .ALIGN  QUAD                    ;                                   [26]
                        00A8   382 DS$BRANCH::                              ;                                   [26]
                0000'   00A8   383          .VECTOR DSX$BRANCH              ; QA branch routine                 [26]
    00000002'EF   17    00AA   384          JMP     DSX$BRANCH+2            ;                                   [26]
                        00B0   385                                          ;                                   [26]
                        00B0   386                                          ;                                   [26]
                        00B0   387
                        00B0   388          .ALIGN  QUAD
                        00B0   389 DS$CVTREG::                              ; CONVERT REGISTER BITS TO ASCII.
                0000'   00B0   390          .VECTOR DSX$CVTREG,^M<>
    00000002'EF   17    00B2   391          JMP     DSX$CVTREG+2
                        00B8   392
                        00B8   393          .ALIGN  QUAD
                        00B8   394 DS$PARSE::                               ; PARSE AN ASCII STRING.
                0000'   00B8   395          .VECTOR DSX$PARSE
    00000002'EF   17    00BA   396          JMP     DSX$PARSE+2
```

```
                        00C0   398          .ALIGN  QUAD
                        00C0   399 DS$ERRSYS::                          ; SYSTEM FATAL ERROR ENTRY POINT.
                0000'   00C0   400          .VECTOR DSX$ERRSYS
00000002'EF      17     00C2   401          JMP     DSX$ERRSYS+2
                        00C8   402
                        00C8   403          .ALIGN  QUAD
                        00C8   404 DS$ERRDEV::                          ; DEVICE FATAL ERROR ENTRY POINT.
                0000'   00C8   405          .VECTOR DSX$ERRDEV
00000002'EF      17     00CA   406          JMP     DSX$ERRDEV+2
                        00D0   407
                        00D0   408          .ALIGN  QUAD
                        00D0   409 DS$ERRHARD::                         ; HARD ERROR ENTRY POINT.
                0000'   00D0   410          .VECTOR DSX$ERRHARD
00000002'EF      17     00D2   411          JMP     DSX$ERRHARD+2
                        C0D8   412
                        00D8   413          .ALIGN  QUAD
                        00D8   414 DS$ERRSOFT::                         ; SOFT ERROR ENTRY POINT.
                0000'   00D8   415          .VECTOR DSX$ERRSOFT
00000002'EF      17     00DA   416          JMP     DSX$ERRSOFT+2
                        00E0   417
                        00E0   418          .ALIGN  QUAD
                        00E0   419 DS$PRINTB::                          ; PRINT BASIC MESSAGE ENTRY POINT.
                0000'   00E0   420          .VECTOR DSX$PRINTB,^M<>
00000002'EF      17     00E2   421          JMP     DSX$PRINTB+2
                        00E8   422
                        00E8   423          .ALIGN  QUAD
                        00E8   424 DS$PRINTX::                          ; PRINT EXTENDED MESSAGE ENTRY POINT.
                0000'   00E8   425          .VECTOR DSX$PRINTX,^M<>
00000002'EF      17     00EA   426          JMP     DSX$PRINTX+2
                        00F0   427
                        00F0   428          .ALIGN  QUAD
                        00F0   429 DS$PRINTF::                          ; PRINT FORCED MESSAGE ENTRY POINT.
                0000'   00F0   430          .VECTOR DSX$PRINTF,^M<>
00000002'EF      17     00F2   431          JMP     DSX$PRINTF+2
                        00F8   432
                        00F8   433          .ALIGN  QUAD
                        00F8   434 DS$PRINTS::                          ; PRINT STATISTICAL MESSAGE ENTRY POINT.
                0000'   00F8   435          .VECTOR DSX$PRINTS,^M<>
00000002'EF      17     00FA   436          JMP     DSX$PRINTS+2
```

```
                     0100   438          .ALIGN  QUAD
                     0100   439 DS$PRINTSIG::                          ; Print signal array routine
                0000' 0100   440          .VECTOR DSX$PRINTSIG,^M<>
    00000002'EF   17  0102   441          JMP     DSX$PRINTSIG+2
                     0108   442
                     0108   443          .ALIGN  QUAD
                     0108   444 DS$ERRPREP::                           ; ERROR IN DEVICE PREPARATION.          [28]
                0000' 0108   445          .VECTOR DSX$ERRPREP          ;                                       [28]
    00000002'EF   17  010A   446          JMP     DSX$ERRPREP+2        ;                                       [28]
                     0110   447
                     0110   448          .ALIGN  QUAD
                     0110   449 DS$SETPRIEXV::                         ; SET PRIMARY EXCEPTION VECTOR  [29]
                0000' 0110   450          .VECTOR DSX$SETPRIEXV,^M<>   ;                                  [29]
    00000002'EF   17  0112   451          JMP     DSX$SETPRIEXV+2      ;                                  [29]
                     0118   452
                     0118   453          .ALIGN  QUAD
                     0118   454 DS$MAPDBGBLOCK::                       ; MAP BLOCK OF MEMORY FOR DEBUGGER[30]
                0000' 0118   455          .VECTOR DSX$MAPDBGBLOCK,^M<> ;                                  [30]
    00000002'EF   17  011A   456          JMP     DSX$MAPDBGBLOCK+2    ;                                  [30]
                     0120   457
```

```
                     0120   459 ;+
                     0120   460 ; 4.2.3          MEMORY MANAGEMENT SERVICES.
                     0120   461 ;-
                     0120   462          .ALIGN  QUAD
                     0120   463 DS$GETBUF::                          ; GET BUFFER ENTRY POINT.
               0000' 0120   464          .VECTOR DSX$GETBUF
00000002'EF    17    0122   465          JMP     DSX$GETBUF+2        ;
                     0128   466
                     0128   467          .ALIGN  QUAD
                     0128   468 DS$RELBUF::                          ; RELEASE BUFFER ENTRY POINT.
               0000' 0128   469          .VECTOR DSX$RELBUF
00000002'EF    17    012A   470          JMP     DSX$RELBUF+2        ;
                     0130   471
                     0130   472          .ALIGN  QUAD
                     0130   473 ; RESERVED ENTRY POINT                ; GET PHYSICAL MEMORY ENTRY POINT.
                0000 0130   474          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF    17    0132   475          JMP     RESRVD_ENTRY        ;
                     0138   476
                     0138   477          .ALIGN  QUAD
                     0138   478 ; RESERVED ENTRY POINT                ; RELEASE PHYSICAL MEMORY ENTRY POINT.
                0000 0138   479          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF    17    013A   480          JMP     RESRVD_ENTRY        ;
                     0140   481
                     0140   482          .ALIGN  QUAD
                     0140   483 ; RESERVED ENTRY POINT                ; RELOCATE PROGRAM ENTRY POINT.
                0000 0140   484          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF    17    0142   485          JMP     RESRVD_ENTRY        ;
                     0148   486
                     0148   487          .ALIGN  QUAD
                     0148   488 ; RESERVED ENTRY POINT                ; PHYSICAL RELOCATE ENTRY POINT.
                0000 0148   489          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF    17    014A   490          JMP     RESRVD_ENTRY        ;
                     0150   491
                     0150   492          .ALIGN  QUAD
                     0150   493 DS$MMON::                            ; MEMORY MANAGEMENT ON ENTRY POINT.
               0000' 0150   494          .VECTOR DSX$MMON,^M<>
00000002'EF    17    0152   495          JMP     DSX$MMON+2          ;
                     0158   496
                     0158   497          .ALIGN  QUAD
                     0158   498 DS$MMOFF::                           ; MEMORY MANAGEMENT OFF ENTRY POINT.
               0000' 0158   499          .VECTOR DSX$MMOFF,^M<>
00000002'EF    17    015A   500          JMP     DSX$MMOFF+2         ;
```

```
                        0160   502 ;+
                        0160   503 ; 4.2.4          I/O SERVICES.
                        0160   504 ;-
                        0160   505          .ALIGN  QUAD
                        0160   506 DS$SETVEC::                          ; SET VECTOR ENTRY POINT.
                0000'   0160   507          .VECTOR DSX$SETVEC
    00000002'EF   17    0162   508          JMP     DSX$SETVEC+2
                        0168   509
                        0168   510          .ALIGN  QUAD
                        0168   511 DS$CLRVEC::                          ; CLEAR VECTOR ENTRY POINT.
                0000'   0168   512          .VECTOR DSX$CLRVEC
    00000002'EF   17    016A   513          JMP     DSX$CLRVEC+2
                        0170   514
                        0170   515          .ALIGN  QUAD
                        0170   516 DS$INITSCB::                         ; INITIALIZE SYSTEM CONTROL BLOCK ENTRY POIN
                0000'   0170   517          .VECTOR DSX$INITSCB
    00000002'EF   17    0172   518          JMP     DSX$INITSCB+2
                        0178   519
                        0178   520          .ALIGN  QUAD
                        0178   521 DS$SETIPL::                          ; SET INTERRUPT PRIORITY LEVEL ENTRY POINT.
                0000'   0178   522          .VECTOR DSX$SETIPL
    00000002'EF   17    017A   523          JMP     DSX$SETIPL+2
                        0180   524
                        0180   525          .ALIGN  QUAD
                        0180   526 DS$CHANNEL::                         ; CHANNEL SERVICES ENTRY POINT.
                0000'   0180   527          .VECTOR DSX$CHANNEL
    00000002'EF   17    0182   528          JMP     DSX$CHANNEL+2
                        0188   529
                        0188   530          .ALIGN  QUAD
                        0188   531 DS$SETMAP::                          ; SET CHANNEL MAPPING ENTRY POINT.
                0000'   0188   532          .VECTOR DSX$SETMAP
    00000002'EF   17    018A   533          JMP     DSX$SETMAP+2
                        0190   534
                        0190   535          .ALIGN  QUAD
                        0190   536 DS$SHOCHAN::
                        0190   537 DS$SHOWCHAN::                        ; SHOW CHANNEL STATUS ENTRY POINT.
                0000'   0190   538          .VECTOR DSX$SHOWCHAN
    00000002'EF   17    0192   539          JMP     DSX$SHOWCHAN+2
                        0198   540
                        0198   541          .ALIGN  QUAD
                        0198   542 DS$LOAD::                           ; ^X10198
                0000'   0198   543          .VECTOR DSX$LOAD
    00000002'EF   17    019A   544          JMP     DSX$LOAD+2
                        01A0   545
                        01A0   546          .ALIGN  QUAD
                        01A0   547 DS$PROBE::                          ; Probe accessibility
                0000'   01A0   548          .VECTOR DSX$PROBE
    00000002'EF   17    01A2   549          JMP     DSX$PROBE+2
                        01A8   550
                        01A8   551          .ALIGN  QUAD
                        01A8   552 DS$ATTACH::                         ; Process ATTACH command line
                0000'   01A8   553          .VECTOR DSX$ATTACH
    00000002'EF   17    01AA   554          JMP     DSX$ATTACH+2
                        01B0   555          .ALIGN  QUAD
                        01B0   556 DS$HELP::                           ; Process HELP command line
                0000'   01B0   557          .VECTOR DSX$HELP
    00000002'EF   17    01B2   558          JMP     DSX$HELP+2
```

```
                        01B8   560          .ALIGN  QUAD
                        01B8   561 DS$FREEDBGSYM::                          ; FREE MEMORY ASSIGNED TO SYMBOL TABLES[31]
                0000'   01B8   562          .VECTOR DSX$FREEDBGSYM,^M<>     ;                                        [31]
    00000002'EF   17    01BA   563          JMP     DSX$FREEDBGSYM+2        ;                                        [31]
                        01C0   564
                        01C0   565          .ALIGN  QUAD
                        01C0   566 DS$LOADPCS::                             ; LOAD THE PCS INTO THE COMET            [32]
                0000'   01C0   567          .VECTOR DSX$LOADPCS,^M<>        ;                                        [32]
    00000002'EF   17    01C2   568          JMP     DSX$LOADPCS+2           ;                                        [32]
                        01C8   569
                        01C8   570          .ALIGN  QUAD
                        01C8   571 DS$GETTERM::                             ; Get terminal characteristics           [34]
                0000'   01C8   572          .VECTOR DSX$GETTERM,^M<>        ;                                        [34]
    00000002'EF   17    01CA   573          JMP     DSX$GETTERM+2           ;                                        [34]
                        01D0   574
                        01D0   575          .ALIGN  QUAD
                        01D0   576 DS$SERVICE_DISPATCHER::                  ;                                        [35]
                0000'   01D0   577          .VECTOR DSX$SERVICE_DISPATCHER  ;                                        [35]
    00000002'EF   17    01D2   578          JMP     DSX$SERVICE_DISPATCHER+2; Common dispatcher for all              [35]
                        01D8   579                                          ; future additional services.            [35]
                        01D8   580
                        01D8   581          .ALIGN  QUAD
                        01D8   582 ; RESERVED ENTRY POINT.
                0000    01D8   583          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    01DA   584          JMP     RESRVD_ENTRY
                        01E0   585
                        01E0   586          .ALIGN  QUAD
                        01E0   587 ; RESERVED ENTRY POINT.
                0000    01E0   588          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    01E2   589          JMP     RESRVD_ENTRY
                        01E8   590
                        01E8   591          .ALIGN  QUAD
                        01E8   592 ; RESERVED ENTRY POINT.
                0000    01E8   593          .WORD   ^M<>                    ; ENTRY MASK.
              0E   11   01EA   594          BRB     RESRVD                  ; LEAPFROG TO RESERVED ENTRY
                        01EC   595 LEAP_BOOT:
    00000000'EF   17    01EC   596          JMP     BOOT                    ; LEAPFROG TO BOOT
                        01F2   597
                        01F2   598 ; RESERVED ENTRY+2      (SAVE MASK IS HIGH ORDER BITS OF BOOT RELATIVE ADDRESS)
              06   11   01F2   599          BRB     RESRVD                  ; LEAPFROG TO RESERVED_ENTRY
                        01F4   600
                        01F4   601 LEAP_APT:
    00000000'EF   17    01F4   602          JMP     APT                     ; LEAPFROG TO APT ENTRY POINT
                        01FA   603 ; RESERVED ENTRY+2      (SAVE MASK IS HIGH ORDER BITS OF APT RELATIVE ADDRESS)
    0000001D'EF   17    01FA   604 RESRVD: JMP     RESRVD_ENTRY
```

```
                     0200   606 ;
                     0200   607 ; STARLET SYSTEM SERVICE VECTORS.
                     0200   608 ;
                     0200   609         .ALIGN  QUAD
                     0200   610 DS$AQ_SYSSRV::
                     0200   611 SYS$QIOW::                                ; ^X80000000
               0000' 0200   612         .VECTOR EXE$QIO
00000002'9F     16   0202   613         JSB     @#EXE$QIO+2
        00000204     0208   614 EXE$QIO2=.-4                             ; FUDGE ADDRESS OF EXE$QIO+2, SEE SYS$QIO
      03 50     E9   0208   615         BLBC    R0, 10$                 ; SKIP WAIT IF QIO ERROR.
       025C'    31   020B   616         BRW     SYS$WAITFR+2            ; WAIT FOR EVENT FLAG.
          04    04   020E   617 10$:    RET
                     020F   618
                     020F   619         .ALIGN  QUAD
                     0210   620 SYS$CALL_HANDL::
 61   04 AE     FA   0210   621         CALLG   4(SP),(R1)              ; CALL EXCEPTION HANDLER
          05    05   0214   622         RSB
                     0215   623
                     0215   624         .ALIGN  QUAD
                     0218   625 ; RESERVED ENTRY POINT.
               0000  0218   626         .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17   021A   627         JMP     RESRVD_ENTRY
                     0220   628
                     0220   629         .ALIGN  QUAD
                     0220   630 ; RESERVED ENTRY POINT.
               0000  0220   631         .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17   0222   632         JMP     RESRVD_ENTRY
                     0228   633
                     0228   634         .ALIGN  QUAD
                     0228   635 ; RESERVED ENTRY POINT.
               0000  0228   636         .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17   022A   637         JMP     RESRVD_ENTRY
                     0230   638
                     0230   639         .ALIGN  QUAD
                     0230   640 ; RESERVED ENTRY POINT.
               0000  0230   641         .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17   0232   642         JMP     RESRVD_ENTRY
                     0238   643
                     0238   644         .ALIGN  QUAD
                     0238   645 SYS$ALLOC::                              ; ^X80000038
               0000' 0238   646         .VECTOR EXE$ALLOC
00000002'EF     17   023A   647         JMP     EXE$ALLOC+2
```

```
                     0240   649          .ALIGN  QUAD
                     0240   650 ; RESERVED ENTRY POINT.                    ; ^X80000040
                0000 0240   651          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0242   652          JMP     RESRVD_ENTRY
                     0248   653
                     0248   654          .ALIGN  QUAD
                     0248   655 SYS$ASCTIM::                               ; ^X80000048
                007C 0248   656          .WORD   ^M<R2,R3,R4,R5,R6>        ; ENTRY MASK.
    00000002'EF   17 024A   657          JMP     EXE$ASCTIM+2
                     0250   658
                     0250   659          .ALIGN  QUAD
                     0250   660 SYS$ASSIGN::                               ; ^X80000050
                0000' 0250   661          .VECTOR EXE$ASSIGN
    00000002'EF   17 0252   662          JMP     EXE$ASSIGN+2
                     C258   663
                     0258   664          .ALIGN  QUAD
                     0258   665 SYS$BINTIM::                               ; ^X80000058
                01FC 0258   666          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8>; ENTRY MASK.
    00000002'EF   17 025A   667          JMP     EXE$BINTIM+2
                     0260   668
                     0260   669          .ALIGN  QUAD
                     0260   670 SYS$CANCEL::                               ; ^X80000060
                0000' 0260   671          .VECTOR EXE$CANCEL
    00000002'EF   17 0262   672          JMP     EXE$CANCEL+2
                     0268   673
                     0268   674          .ALIGN  QUAD
                     0268   675 SYS$CANTIM::                               ; ^X80000068
                0000' 0268   676          .VECTOR EXE$CANTIM
    00000002'EF   17 026A   677          JMP     EXE$CANTIM+2
                     0270   678
                     0270   679          .ALIGN  QUAD
                     0270   680 ; RESERVED ENTRY POINT.
                0000 0270   681          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0272   682          JMP     RESRVD_ENTRY
                     0278   683
                     0278   684          .ALIGN  QUAD
                     0278   685 ; RESERVED ENTRY POINT.
                0000 0278   686          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 027A   687          JMP     RESRVD_ENTRY
```

```
                     0280   689          .ALIGN  QUAD
                     0280   690 ; RESERVED ENTRY POINT.                    ; ^X80000080
                0000 0280   691          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0282   692          JMP     RESRVD_ENTRY
                     0288   693
                     0288   694          .ALIGN  QUAD
                     0288   695 ; RESERVED ENTRY POINT.
                0000 0288   696          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 028A   697          JMP     RESRVD_ENTRY
                     0290   698
                     0290   699          .ALIGN  QUAD
                     0290   700 ; RESERVED ENTRY POINT.
                0000 0290   701          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0292   702          JMP     RESRVD_ENTRY
                     C298   703
                     0298   704          .ALIGN  QUAD
                     0298   705 SYS$CLREF::                                ; ^X80000098
                0000' 0298   706          .vector exe$clref
    00000002'EF   17 029A   707          jmp     exe$clref+2
                     02A0   708
                     02A0   709          .ALIGN  QUAD
                     02A0   710 SYS$CNTREG::                               ; ^X800000A0
                0000' 02A0   711          .VECTOR EXE$CNTREG,^M<>
    00000002'EF   17 02A2   712          JMP     EXE$CNTREG+2
                     02A8   713
                     02A8   714          .ALIGN  QUAD
                     02A8   715 ; RESERVED ENTRY POINT.
                0000 02A8   716          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 02AA   717          JMP     RESRVD_ENTRY
                     02B0   718
                     02B0   719          .ALIGN  QUAD
                     02B0   720 ; RESERVED ENTRY POINT.
                0000 02B0   721          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 02B2   722          JMP     RESRVD_ENTRY
                     02B8   723
                     02B8   724          .ALIGN  QUAD
                     02B8   725 ; RESERVED ENTRY POINT.
                0000 02B8   726          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 02BA   727          JMP     RESRVD_ENTRY
```

```
                        02C0   729          .ALIGN  QUAD
                        02C0   730 ; RESERVED ENTRY POINT.                  ; ^X800000C0
                0000    02C0   731          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    02C2   732          JMP     RESRVD_ENTRY
                        02C8   733
                        02C8   734          .ALIGN  QUAD
                        02C8   735 ; RESERVED ENTRY POINT
                0000    02C8   736          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    02CA   737          JMP     RESRVD_ENTRY
                        02D0   738
                        02D0   739          .ALIGN  QUAD
                        02D0   740 ; RESERVED ENTRY POINT.
                0000    02D0   741          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    02D2   742          JMP     RESRVD_ENTRY
                        C2D8   743
                        02D8   744          .ALIGN  QUAD
                        02D8   745 SYS$DALLOC::                             ; ^X800000D8
                0000'   02D8   746          .VECTOR EXE$DALLOC
    00000002'EF   17    02DA   747          JMP     EXE$CANCEL+2
                        02E0   748
                        02E0   749          .ALIGN  QUAD
                        02E0   750 SYS$DASSGN::                             ; ^X800000E0
                0000'   02E0   751          .VECTOR EXE$DASSGN
    00000002'EF   17    02E2   752          JMP     EXE$DASSGN+2
                        02E8   753
                        02E8   754          .ALIGN  QUAD
                        02E8   755 ; RESERVED ENTRY POINT.
                0000    02E8   756          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    02EA   757          JMP     RESRVD_ENTRY
                        02F0   758
                        02F0   759          .ALIGN  QUAD
                        02F0   760 ; RESERVED ENTRY POINT.
                0000    02F0   761          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    02F2   762          JMP     RESRVD_ENTRY
                        02F8   763
                        02F8   764          .ALIGN  QUAD
                        02F8   765 ; RESERVED ENTRY POINT.
                0000    02F8   766          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    02FA   767          JMP     RESRVD_ENTRY
```

```
                     0300   769          .ALIGN  QUAD
                     0300   770 ; RESERVED ENTRY POINT.                    ; ^X80001000
                0000 0300   771          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0302   772          JMP     RESRVD_ENTRY
                     0308   773
                     0308   774          .ALIGN  QUAD
                     0308   775 ; RESERVED ENTRY POINT.
                0000 0308   776          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 030A   777          JMP     RESRVD_ENTRY
                     0310   778
                     0310   779          .ALIGN  QUAD
                     0310   780 ; RESERVED ENTRY POINT
                0000 0310   781          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0312   782          JMP     RESRVD_ENTRY
                     0318   783
                     0318   784          .ALIGN  QUAD
                     0318   785 ; RESERVED ENTRY POINT.
                0000 0318   786          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 031A   787          JMP     RESRVD_ENTRY
                     0320   788
                     0320   789          .ALIGN  QUAD
                     0320   790 ; RESERVED ENTRY POINT.
                0000 0320   791          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0322   792          JMP     RESRVD_ENTRY
                     0328   793
                     0328   794          .ALIGN  QUAD
                     0328   795 ; RESERVED ENTRY POINT.
                0000 0328   796          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 032A   797          JMP     RESRVD_ENTRY
                     0330   798
                     0330   799          .ALIGN  QUAD
                     0330   800 ; RESERVED ENTRY POINT.
                0000 0330   801          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0332   802          JMP     RESRVD_ENTRY
                     0338   803
                     0338   804          .ALIGN  QUAD
                     0338   805 ; RESERVED ENTRY POINT.
                0000 0338   806          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 033A   807          JMP     RESRVD_ENTRY
```

```
                        0340   809          .ALIGN  QUAD
                        0340   810 SYS$EXIT::                                  ; ^X80000140
                0000    0340   811          .WORD   ^M<>                       ; ENTRY MASK.
                  04    0342   812          RET
                        0343   813
                        0343   814          .ALIGN  QUAD
                        0348   815 SYS$EXPREG::                                ; ^X80000148
                0000'   0348   816          .VECTOR EXE$EXPREG,^M<>
    00000002'EF   17    034A   817          JMP     EXE$EXPREG+2
                        0350   818
                        0350   819          .ALIGN  QUAD
                        0350   820 SYS$FAO::                                   ; ^X80000150
                0FFC    0350   821          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; ENTRY MASK.
    00000002'EF   17    0352   822          JMP     EXE$FAO+2
                        0358   823
                        0358   824          .ALIGN  QUAD
                        0358   825 SYS$FAOL::                                  ; ^X80000158
                0FFC    0358   826          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; ENTRY MASK.
    00000002'EF   17    035A   827          JMP     EXE$FAOL+2
                        0360   828
                        0360   829          .ALIGN  QUAD
                        0360   830 ; RESERVED ENTRY POINT.
                0000    0360   831          .WORD   ^M<>                       ; ENTRY MASK.
    0000001D'EF   17    0362   832          JMP     RESRVD_ENTRY
                        0368   833
                        0368   834          .ALIGN  QUAD
                        0368   835 ; RESERVED ENTRY POINT.
                0000    0368   836          .WORD   ^M<>                       ; ENTRY MASK.
    0000001D'EF   17    036A   837          JMP     RESRVD_ENTRY
                        0370   838
                        0370   839          .ALIGN  QUAD
                        0370   840 ; RESERVED ENTRY POINT.
                0000    0370   841          .WORD   ^M<>                       ; ENTRY MASK.
    0000001D'EF   17    0372   842          JMP     RESRVD_ENTRY
                        0378   843
                        0378   844          .ALIGN  QUAD
                        0378   845 SYS$GETTIM::                                ; ^X80000178
                0000    0378   846          .WORD   ^M<>                       ; ENTRY MASK.
    00000002'EF   17    037A   847          JMP     EXE$GETTIM+2
```

```
                        0380   849          .ALIGN  QUAD
                        0380   850 SYS$GTCHAN::                                ; ^X80000180
                    0000' 0380   851          .VECTOR EXE$GTCHAN
        00000002'EF   17  0382   852          JMP     EXE$GTCHAN+2
                        0388   853
                        0388   854          .ALIGN  QUAD
                        0388   855 SYS$HIBER::                                 ; ^X80000188
                    0000' 0388   856          .VECTOR EXE$HIBER
        00000002'EF   17  038A   857          JMP     EXE$HIBER+2
                        0390   858
                        0390   859          .ALIGN  QUAD
                        0390   860 ; RESERVED ENTRY POINT.
                    0000  0390   861          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF   17  0392   862          JMP     RESRVD_ENTRY
                        0398   863
                        0398   864          .ALIGN  QUAD
                        0398   865 ; RESERVED ENTRY POINT.
                    0000  0398   866          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF   17  039A   867          JMP     RESRVD_ENTRY
                        03A0   868
                        03A0   869          .ALIGN  QUAD
                        03A0   870 SYS$LKWSET::                                ; ^X800001A0
                    0000  03A0   871          .WORD   0                          ; Entry mask
             50   01    D0  03A2   872          MOVL    #1,R0                      ; Indicate success
                        04  03A5   873          RET                                ; Return
                        03A6   874
                        03A6   875          .ALIGN  QUAD
                        03A8   876 ; RESERVED ENTRY POINT.
                    0000  03A8   877          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF   17  03AA   878          JMP     RESRVD_ENTRY
                        03B0   879
                        03B0   880          .ALIGN  QUAD
                        03B0   881 ; RESERVED ENTRY POINT.
                    0000  03B0   882          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF   17  03B2   883          JMP     RESRVD_ENTRY
                        03B8   884
                        03B8   885          .ALIGN  QUAD
                        03B8   886 SYS$NUMTIM::                                ; ^X800001B8
                    00FC  03B8   887          .WORD   ^M<R2,R3,R4,R5,R6,R7>      ; ENTRY MASK.
        00000002'EF   17  03BA   888          JMP     EXE$NUMTIM+2
```

```
                        03C0   890          .ALIGN  QUAD
                        03C0   891 ; RESERVED ENTRY POINT.                  ; ^X800001C0
                  0000  03C0   892          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03C2   893          JMP     RESRVD_ENTRY
                        03C8   894
                        03C8   895          .ALIGN  QUAD
                        03C8   896 SYS$QIO::                               ; ^X800001C8
                  0000' 03C8   897          .VECTOR EXE$QIO
            FE36 DF   16  03CA   898          JSB     @EXE$QIO2
                      04  03CE   899          RET
                        03CF   900
                        03CF   901          .ALIGN  QUAD
                        03D0   902 SYS$READEF::                            ; ^X800001D0
                  0000' 03DC   903          .vector exe$readef
      00000002'EF   17  03D2   904          jmp     exe$readef+2
                        03D8   905
                        03D8   906          .ALIGN  QUAD
                        03D8   907 ; RESERVED ENTRY POINT.
                  0000  03D8   908          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03DA   909          JMP     RESRVD_ENTRY
                        03E0   910
                        03E0   911          .ALIGN  QUAD
                        03E0   912 ; RESERVED ENTRY POINT.
                  0000  03E0   913          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03E2   914          JMP     RESRVD_ENTRY
                        03E8   915
                        03E8   916          .ALIGN  QUAD
                        03E8   917 ; RESERVED ENTRY POINT.
                  0000  03E8   918          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03EA   919          JMP     RESRVD_ENTRY
                        03F0   920
                        03F0   921          .ALIGN  QUAD
                        03F0   922 ; RESERVED ENTRY POINT.
                  0000  03F0   923          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03F2   924          JMP     RESRVD_ENTRY
                        03F8   925
                        03F8   926          .ALIGN  QUAD
                        03F8   927 SYS$SETAST::                            ; ^X800001F8
                  0000' 03F8   928          .VECTOR EXE$SETAST               ; Entry mask
      00000002'EF   17  03FA   929          JMP     EXE$SETAST+2
```

```
                        0400   931          .ALIGN  QUAD
                        0400   932 SYS$SETEF::                          ; ^X80000200
                  0000' 0400   933          .vector exe$setef
      00000002'EF   17  0402   934          jmp     exe$setef+2
                        0408   935
                        0408   936          .ALIGN  QUAD
                        0408   937 ; RESERVED ENTRY POINT.
                  0000  0408   938          .WORD   ^M<>                   ; ENTRY MASK.
      0000001D'EF   17  040A   939          JMP     RESRVD_ENTRY
                        0410   940
                        0410   941          .ALIGN  QUAD
                        0410   942 ; RESERVED ENTRY POINT.
                  0000  0410   943          .WORD   ^M<>                   ; ENTRY MASK.
      0000001D'EF   17  0412   944          JMP     RESRVD_ENTRY
                        C418   945
                        0418   946          .ALIGN  QUAD
                        0418   947 ; RESERVED ENTRY POINT.
                  0000  0418   948          .WORD   ^M<>                   ; ENTRY MASK.
      0000001D'EF   17  041A   949          JMP     RESRVD_ENTRY
                        0420   950
                        0420   951          .ALIGN  QUAD
                        0420   952 SYS$SETIMR::                          ; ^X80000220
                  0000' 0420   953          .VECTOR EXE$SETIMR
      00000002'EF   17  0422   954          JMP     EXE$SETIMR+2
                        0428   955
                        0428   956          .ALIGN  QUAD
                        0428   957 SYS$SETPRI::                          ; ^X80000228
                  0000  0428   958          .WORD   0                      ; Entry mask
          50    01    D0  042A   959          MOVL    #1,R0                  ; Indicate success
                    04  042D   960          RET                            ; Return
                        042E   961
                        042E   962          .ALIGN  QUAD
                        0430   963 SYS$SETPRT::                          ; ^X80000230
                  0000' 0430   964          .VECTOR EXE$SETPRT,^M<>        ; ENTRY MASK.
    00000002'EF   17  0432   965          JMP     EXE$SETPRT+2
                        0438   966
                        0438   967          .ALIGN  QUAD
                        0438   968 SYS$SETRWM::                          ; ^X80000238
                  0000  0438   969          .WORD   0                      ; Entry mask
          50    01    D0  043A   970          MOVL    #1,R0                  ; Indicate success
                    04  043D   971          RET                            ; Return
```

```
                        043E   973          .ALIGN  QUAD
                        0440   974 ; RESERVED ENTRY POINT.                ; ^X80000240
                  0000  0440   975          .WORD   ^M<>                  ; ENTRY MASK.
      0000001D'EF   17  0442   976          JMP     RESRVD_ENTRY
                        0448   977
                        0448   978          .ALIGN  QUAD
                        0448   979 ; RESERVED ENTRY POINT.
                  0000  0448   980          .WORD   ^M<>                  ; ENTRY MASK.
      0000001D'EF   17  044A   981          JMP     RESRVD_ENTRY
                        0450   982
                        0450   983          .ALIGN  QUAD
                        0450   984 ; RESERVED ENTRY POINT.
                  0000  0450   985          .WORD   ^M<>                  ; ENTRY MASK.
      0000001D'EF   17  0452   986          JMP     RESRVD_ENTRY
                        0458   987
                        0458   988          .ALIGN  QUAD
                        0458   989 SYS$TRNLOG::                           ; ^X80000258
                  0000  0458   990          .WORD   ^M<>                  ; ENTRY MASK.
      0000001D'EF   17  045A   991          JMP     RESRVD_ENTRY
                        0460   992
                        0460   993          .ALIGN  QUAD
                        0460   994 SYS$ULKPAG::                           ; ^X80000260
                  0000  0460   995          .WORD   0                     ; Entry mask
           50   01    D0  0462   996          MOVL    #1,R0                 ; Indicate success
                        04  0465   997          RET                           ; Return
                        0466   998
                        0466   999          .ALIGN  QUAD
                        0468  1000 SYS$ULWSET::                           ; ^X80000268
                  0000  0468  1001          .WORD   0                     ; Entry mask
           50   01    D0  046A  1002          MOVL    #1,R0                 ; Indicate success
                        04  046D  1003          RET                           ; Return
                        046E  1004
                        046E  1005          .ALIGN  QUAD
                        0470  1006 SYS$UNWIND::
                  0000' 0470  1007          .VECTOR EXE$UNWIND
      00000002'EF   17  0472  1008          JMP     EXE$UNWIND+2
                        0478  1009
                        0478  1010          .ALIGN  QUAD
                        0478  1011 SYS$WAITFR::                           ; ^X80000278
                  0000' 0478  1012          .vector exe$waitfr
      00000002'EF   17  047A  1013          jmp     exe$waitfr+2
```

```
                        0480  1015          .ALIGN  QUAD
                        0480  1016 SYS$WAKE::                                ; ^X80000280
                  0000' 0480  1017          .VECTOR EXE$WAKE
00000002'EF       17    0482  1018          JMP     EXE$WAKE+2
                        0488  1019
                        0488  1020          .ALIGN  QUAD
                        0488  1021 SYS$WFLAND::                              ; ^X80000288
                  0000' 0488  1022          .vector exe$wfland
00000002'EF       17    048A  1023          jmp     exe$wfland+2
                        0490  1024
                        0490  1025          .ALIGN  QUAD
                        0490  1026 SYS$WFLOR::                               ; ^X80000290
                  0000' 0490  1027          .vector exe$wflor
00000002'EF       17    0492  1028          jmp     exe$wflor+2
                        0498  1029
                        0498  1030          .ALIGN  QUAD
                        0498  1031 ; RESERVED ENTRY POINT.
                  0000  0498  1032          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF       17    049A  1033          JMP     RESRVD_ENTRY
                        04A0  1034
                        04A0  1035          .ALIGN  QUAD
                        04A0  1036 ; RESERVED ENTRY POINT.
                  0000  04A0  1037          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF       17    04A2  1038          JMP     RESRVD_ENTRY
                        04A8  1039
                        04A8  1040          .ALIGN  QUAD
                        04A8  1041 ; RESERVED ENTRY POINT.
                  0000  04A8  1042          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF       17    04AA  1043          JMP     RESRVD_ENTRY
                        04B0  1044
                        04B0  1045          .ALIGN  QUAD
                        04B0  1046 ; RESERVED ENTRY POINT.
                  0000  04B0  1047          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF       17    04B2  1048          JMP     RESRVD_ENTRY
                        04B8  1049
                        04B8  1050          .ALIGN  QUAD
                        04B8  1051 ; RESERVED ENTRY POINT.
                  0000  04B8  1052          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF       17    04BA  1053          JMP     RESRVD_ENTRY
```

```
                        04C0  1055          .ALIGN  QUAD
                        04C0  1056 ; RESERVED ENTRY POINT.                  ; ^X800002C0
                0000    04C0  1057          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04C2  1058          JMP     RESRVD_ENTRY
                        04C8  1059
                        04C8  1060          .ALIGN  QUAD
                        04C8  1061 SYS$GETCHN::                             ; ^X800002C8
                0000'   04C8  1062          .VECTOR EXE$GTCHAN
    00000002'EF   17    04CA  1063          JMP     EXE$GTCHAN+2
                        04D0  1064
                        04D0  1065          .ALIGN  QUAD
                        04D0  1066 ; RESERVED ENTRY POINT.
                0000    04D0  1067          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04D2  1068          JMP     RESRVD_ENTRY
                        C4D8  1069
                        04D8  1070          .ALIGN  QUAD
                        04D8  1071 ; RESERVED ENTRY POINT.
                0000    04D8  1072          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04DA  1073          JMP     RESRVD_ENTRY
                        04E0  1074
                        04E0  1075          .ALIGN  QUAD
                        04E0  1076 ; RESERVED ENTRY POINT.
                0000    04E0  1077          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04E2  1078          JMP     RESRVD_ENTRY
                        04E8  1079
                        04E8  1080          .ALIGN  QUAD
                        04E8  1081 ; RESERVED ENTRY POINT.
                0000    04E8  1082          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04EA  1083          JMP     RESRVD_ENTRY
                        04F0  1084
                        04F0  1085          .ALIGN  QUAD
                        04F0  1086 ; RESERVED ENTRY POINT.
                0000    04F0  1087          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04F2  1088          JMP     RESRVD_ENTRY
                        04F8  1089
                        04F8  1090          .ALIGN  QUAD
                        04F8  1091 ; RESERVED ENTRY POINT.
                0000    04F8  1092          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04FA  1093          JMP     RESRVD_ENTRY
```

```
                        0500  1095          .ALIGN  QUAD
                        0500  1096 ; RESERVED ENTRY POINT.                    ; ^X80000300
                0000    0500  1097          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0502  1098          JMP     RESRVD_ENTRY
                        0508  1099
                        0508  1100          .ALIGN  QUAD
                        0508  1101 ; RESERVED ENTRY POINT.
                0000    0508  1102          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    050A  1103          JMP     RESRVD_ENTRY
                        0510  1104
                        0510  1105          .ALIGN  QUAD
                        0510  1106 ; RESERVED ENTRY POINT.
                C000    0510  1107          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0512  1108          JMP     RESRVD_ENTRY
                        0518  1109
                        0518  1110          .ALIGN  QUAD
                        0518  1111 ; RESERVED ENTRY POINT.
                0000    0518  1112          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    051A  1113          JMP     RESRVD_ENTRY
                        0520  1114
                        0520  1115          .ALIGN  QUAD
                        0520  1116 ; RESERVED ENTRY POINT.
                0000    0520  1117          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0522  1118          JMP     RESRVD_ENTRY
                        0528  1119
                        0528  1120          .ALIGN  QUAD
                        0528  1121 ; RESERVED ENTRY POINT.
                0000    0528  1122          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    052A  1123          JMP     RESRVD_ENTRY
                        0530  1124
                        0530  1125          .ALIGN  QUAD
                        0530  1126 ; RESERVED ENTRY POINT.
                0000    0530  1127          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0532  1128          JMP     RESRVD_ENTRY
                        0538  1129
                        0538  1130          .ALIGN  QUAD
                        0538  1131 ; RESERVED ENTRY POINT.
                0000    0538  1132          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    053A  1133          JMP     RESRVD_ENTRY
```

```
                        0540  1135          .ALIGN  QUAD
                        0540  1136 ; RESERVED ENTRY POINT                    ; ^X80000340
                0000    0540  1137          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    0542  1138          JMP     RESRVD_ENTRY
                        0548  1139
                        0548  1140          .ALIGN  QUAD
                        0548  1141 ; RESERVED ENTRY POINT.
                0000    0548  1142          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    054A  1143          JMP     RESRVD_ENTRY
                        0550  1144
                        0550  1145          .ALIGN  QUAD
                        0550  1146 ; RESERVED ENTRY POINT.
                0000    0550  1147          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    0552  1148          JMP     RESRVD_ENTRY
                        0558  1149
                        0558  1150          .ALIGN  QUAD
                        0558  1151 ; RESERVED ENTRY POINT.
                0000    0558  1152          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    055A  1153          JMP     RESRVD_ENTRY
                        0560  1154
                        0560  1155          .ALIGN  QUAD
                        0560  1156 ; RESERVED ENTRY POINT.
                0000    0560  1157          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    0562  1158          JMP     RESRVD_ENTRY
                        0568  1159
                        0568  1160          .ALIGN  QUAD
                        0568  1161 ; RESERVED ENTRY POINT.
                0000    0568  1162          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    056A  1163          JMP     RESRVD_ENTRY
                        0570  1164
                        0570  1165          .ALIGN  QUAD
                        0570  1166 ; RESERVED ENTRY POINT.
                0000    0570  1167          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    0572  1168          JMP     RESRVD_ENTRY
                        0578  1169
                        0578  1170          .ALIGN  QUAD
                        0578  1171 ; RESERVED ENTRY POINT.
                0000    0578  1172          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    057A  1173          JMP     RESRVD_ENTRY
```

```
                        0580  1175          .ALIGN  QUAD
                        0580  1176 SYS$GET::                                    ; ^X80000380
                0000'   0580  1177          .VECTOR RMS$GET
    00000002'EF   17    0582  1178          JMP     RMS$GET+2
                        0588  1179
                        0588  1180          .ALIGN  QUAD
                        0588  1181 ; RESERVED ENTRY POINT.
                0000    0588  1182          .WORD   ^M<>                        ; ENTRY MASK.
    0000001D'EF   17    058A  1183          JMP     RESRVD_ENTRY
                        0590  1184
                        0590  1185          .ALIGN  QUAD
                        0590  1186 SYS$READ::
                0000'   0590  1187          .VECTOR RMS$READ
    00000002'EF   17    0592  1188          JMP     RMS$READ+2                  ; ^X80000390
                        0598  1189
                        0598  1190          .ALIGN  QUAD
                        0598  1191 ; RESERVED ENTRY POINT.
                0000    0598  1192          .WORD   ^M<>                        ; ENTRY MASK.
    0000001D'EF   17    059A  1193          JMP     RESRVD_ENTRY
                        05A0  1194
                        05A0  1195          .ALIGN  QUAD
                        05A0  1196 ; RESERVED ENTRY POINT.
                0000    05A0  1197          .WORD   ^M<>                        ; ENTRY MASK.
    0000001D'EF   17    05A2  1198          JMP     RESRVD_ENTRY
                        05A8  1199
                        05A8  1200          .ALIGN  QUAD
                        05A8  1201 ; RESERVED ENTRY POINT.
                0000    05A8  1202          .WORD   ^M<>                        ; ENTRY MASK.
    0000001D'EF   17    05AA  1203          JMP     RESRVD_ENTRY
                        05B0  1204
                        05B0  1205          .ALIGN  QUAD
                        05B0  1206 ; RESERVED ENTRY POINT.
                0000    05B0  1207          .WORD   ^M<>                        ; ENTRY MASK.
    0000001D'EF   17    05B2  1208          JMP     RESRVD_ENTRY
                        05B8  1209
                        05E8  1210          .ALIGN  QUAD
                        0538  1211 SYS$CLOSE::
                0000'   05B8  1212          .VECTOR RMS$CLOSE
    00000002'EF   17    05BA  1213          JMP     RMS$CLOSE+2
```

```
                        05C0  1215          .ALIGN  QUAD
                        05C0  1216 SYS$CONNECT::                            ; ^X800003C0
                0000'   05C0  1217          .VECTOR RMS$CONNECT
    00000002'EF   17    05C2  1218          JMP     RMS$CONNECT+2
                        05C8  1219
                        05C8  1220          .ALIGN  QUAD
                        05C8  1221 ; RESERVED ENTRY POINT.
                0000    05C8  1222          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05CA  1223          JMP     RESRVD_ENTRY
                        05D0  1224
                        05D0  1225          .ALIGN  QUAD
                        05D0  1226 SYS$DISCONNECT::                                 ; ^X800003D0
                0000'   05D0  1227          .VECTOR RMS$DISCONNECT
    00000002'EF   17    05D2  1228          JMP     RMS$DISCONNECT+2
                        C5D8  1229
                        05D8  1230          .ALIGN  QUAD
                        05D8  1231 ; RESERVED ENTRY POINT.
                0000    05D8  1232          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05DA  1233          JMP     RESRVD_ENTRY
                        05E0  1234
                        05E0  1235          .ALIGN  QUAD
                        05E0  1236 ; RESERVED ENTRY POINT.
                0000    05E0  1237          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05E2  1238          JMP     RESRVD_ENTRY
                        05E8  1239
                        05E8  1240          .ALIGN  QUAD
                        05E8  1241 ; RESERVED ENTRY POINT.
                0000    05E8  1242          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05EA  1243          JMP     RESRVD_ENTRY
                        05F0  1244
                        05F0  1245          .ALIGN  QUAD
                        05F0  1246 ; RESERVED ENTRY POINT.
                0000    05F0  1247          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05F2  1248          JMP     RESRVD_ENTRY
                        05F8  1249
                        05F8  1250          .ALIGN  QUAD
                        05F8  1251 ; RESERVED ENTRY POINT.
                0000    05F8  1252          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05FA  1253          JMP     RESRVD_ENTRY
```

```
                     0600  1255          .ALIGN  QUAD
                     0600  1256 ; RESERVED ENTRY POINT.                ; ^X80000400
                0000 0600  1257          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 0602  1258          JMP     RESRVD_ENTRY
                     0608  1259
                     0608  1260          .ALIGN  QUAD
                     0608  1261 SYS$OPEN::                             ; ^X80000408
                0000'0608  1262          .VECTOR RMS$OPEN
    00000002'EF   17 060A  1263          JMP     RMS$OPEN+2
                     0610  1264
                     0610  1265
                     0610  1266          .ALIGN  QUAD
                     0610  1267 ; RESERVED ENTRY POINT.
                0000 0610  1268          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 0612  1269          JMP     RESRVD_ENTRY
                     0618  1270
                     0618  1271          .ALIGN  QUAD
                     0618  1272 ; RESERVED ENTRY POINT.
                0000 0618  1273          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 061A  1274          JMP     RESRVD_ENTRY
                     0620  1275
                     0620  1276          .ALIGN  QUAD
                     0620  1277 ; RESERVED ENTRY POINT.
                0000 0620  1278          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 0622  1279          JMP     RESRVD_ENTRY
                     0628  1280
                     0628  1281          .ALIGN  QUAD
                     0628  1282 ; RESERVED ENTRY POINT.
                0000 0628  1283          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 062A  1284          JMP     RESRVD_ENTRY
                     0630  1285
                     0630  1286          .ALIGN  QUAD
                     0630  1287 ; RESERVED ENTRY POINT.
                0000 0630  1288          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 0632  1289          JMP     RESRVD_ENTRY
                     0638  1290
                     0638  1291          .ALIGN  QUAD
                     0638  1292 ; RESERVED ENTRY POINT.
                0000 0638  1293          .WORD   ^M<>                  ; ENTRY MASK.
    0000001D'EF   17 063A  1294          JMP     RESRVD_ENTRY
```

```
                        0640  1296          .ALIGN  QUAD
                        0640  1297 ; RESERVED ENTRY POINT.                ; ^X80000440
                0000    0640  1298          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     0642  1299          JMP     RESRVD_ENTRY
                        0648  1300
                        0648  1301          .ALIGN  QUAD
                        0648  1302 ; RESERVED ENTRY POINT.
                0000    0648  1303          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     064A  1304          JMP     RESRVD_ENTRY
                        0650  1305
                        0650  1306          .ALIGN  QUAD
                        0650  1307 ; RESERVED ENTRY POINT.
                0000    0650  1308          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     0652  1309          JMP     RESRVD_ENTRY
                        0658  1310
                        0658  1311          .ALIGN  QUAD
                        0658  1312 ; RESERVED ENTRY POINT.
                0000    0658  1313          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     065A  1314          JMP     RESRVD_ENTRY
                        0660  1315
                        0660  1316          .ALIGN  QUAD
                        0660  1317 ; RESERVED ENTRY POINT.
                0000    0660  1318          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     0662  1319          JMP     RESRVD_ENTRY
                        0668  1320
                        0668  1321          .ALIGN  QUAD
                        0668  1322 ; RESERVED ENTRY POINT.
                0000    0668  1323          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     066A  1324          JMP     RESRVD_ENTRY
                        0670  1325
                        0670  1326          .ALIGN  QUAD
                        0670  1327 ; RESERVED ENTRY POINT.
                0000    0670  1328          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     0672  1329          JMP     RESRVD_ENTRY
                        0678  1330
                        0678  1331          .ALIGN  QUAD
                        0678  1332 ; RESERVED ENTRY POINT.
                0000    0678  1333          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF      17     067A  1334          JMP     RESRVD_ENTRY
```

```
                    0680  1336          .ALIGN  QUAD
                    0680  1337 ; RESERVED ENTRY POINT.                ; ^X80000480
              0000  0680  1338          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   0682  1339          JMP     RESRVD_ENTRY
                    0688  1340
                    0688  1341          .ALIGN  QUAD
                    0688  1342 ; RESERVED ENTRY POINT.
              0000  0688  1343          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   068A  1344          JMP     RESRVD_ENTRY
                    0690  1345
                    0690  1346          .ALIGN  QUAD
                    0690  1347 ; RESERVED ENTRY POINT.
              0000  0690  1348          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   0692  1349          JMP     RESRVD_ENTRY
                    C698  1350
                    0698  1351          .ALIGN  QUAD
                    0698  1352 ; RESERVED ENTRY POINT.
              0000  0698  1353          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   069A  1354          JMP     RESRVD_ENTRY
                    06A0  1355
                    06A0  1356          .ALIGN  QUAD
                    06A0  1357 ; RESERVED ENTRY POINT.
              0000  06A0  1358          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   06A2  1359          JMP     RESRVD_ENTRY
                    06A8  1360
                    06A8  1361          .ALIGN  QUAD
                    06A8  1362 ; RESERVED ENTRY POINT.
              0000  06A8  1363          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   06AA  1364          JMP     RESRVD_ENTRY
                    06B0  1365
                    06B0  1366          .ALIGN  QUAD
                    06B0  1367 ; RESERVED ENTRY POINT.
              0000  06B0  1368          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   06B2  1369          JMP     RESRVD_ENTRY
                    06B8  1370
                    06B8  1371          .ALIGN  QUAD
                    06B8  1372 ; RESERVED ENTRY POINT.
              0000  06B8  1373          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   06BA  1374          JMP     RESRVD_ENTRY
```

```
                        06C0  1376          .ALIGN  QUAD
                        06C0  1377 ; RESERVED ENTRY POINT.                  ; ^X800004C0
                  0000  06C0  1378          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06C2  1379          JMP     RESRVD_ENTRY
                        06C8  1380
                        06C8  1381          .ALIGN  QUAD
                        06C8  1382 ; RESERVED ENTRY POINT.
                  0000  06C8  1383          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06CA  1384          JMP     RESRVD_ENTRY
                        06D0  1385
                        06D0  1386          .ALIGN  QUAD
                        06D0  1387 ; RESERVED ENTRY POINT.
                  0000  06D0  1388          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06D2  1389          JMP     RESRVD_ENTRY
                        06D8  1390
                        06D8  1391          .ALIGN  QUAD
                        06D8  1392 ; RESERVED ENTRY POINT.
                  0000  06D8  1393          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06DA  1394          JMP     RESRVD_ENTRY
                        06E0  1395
                        06E0  1396          .ALIGN  QUAD
                        06E0  1397 ; RESERVED ENTRY POINT.
                  0000  06E0  1398          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06E2  1399          JMP     RESRVD_ENTRY
                        06E8  1400
                        06E8  1401          .ALIGN  QUAD
                        06E8  1402 ; RESERVED ENTRY POINT.
                  0000  06E8  1403          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06EA  1404          JMP     RESRVD_ENTRY
                        06F0  1405
                        06F0  1406          .ALIGN  QUAD
                        06F0  1407 ; RESERVED ENTRY POINT.
                  0000  06F0  1408          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06F2  1409          JMP     RESRVD_ENTRY
                        06F8  1410
                        06F8  1411          .ALIGN  QUAD
                        06F8  1412 ; RESERVED ENTRY POINT.
                  0000  06F8  1413          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  06FA  1414          JMP     RESRVD_ENTRY
```

```
                    0700  1416          .ALIGN  QUAD
                    0700  1417 ; RESERVED ENTRY POINT.                ; ^X80000500
              0000  0700  1418          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   0702  1419          JMP     RESRVD_ENTRY
                    0708  1420
                    0708  1421          .ALIGN  QUAD
                    0708  1422 ; RESERVED ENTRY POINT.
              0000  0708  1423          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   070A  1424          JMP     RESRVD_ENTRY
                    0710  1425
                    0710  1426          .ALIGN  QUAD
                    0710  1427 ; RESERVED ENTRY POINT.
              0000  0710  1428          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   0712  1429          JMP     RESRVD_ENTRY
                    0718  1430
                    0718  1431          .ALIGN  QUAD
                    0718  1432 ; RESERVED ENTRY POINT.
              0000  0718  1433          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   071A  1434          JMP     RESRVD_ENTRY
                    0720  1435
                    0720  1436          .ALIGN  QUAD
                    0720  1437 ; RESERVED ENTRY POINT.
              0000  0720  1438          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   0722  1439          JMP     RESRVD_ENTRY
                    0728  1440
                    0728  1441          .ALIGN  QUAD
                    0728  1442 ; RESERVED ENTRY POINT.
              0000  0728  1443          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   072A  1444          JMP     RESRVD_ENTRY
                    0730  1445
                    0730  1446          .ALIGN  QUAD
                    0730  1447 ; RESERVED ENTRY POINT.
              0000  0730  1448          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   0732  1449          JMP     RESRVD_ENTRY
                    0738  1450
                    0738  1451          .ALIGN  QUAD
                    0738  1452 ; RESERVED ENTRY POINT.
              0000  0738  1453          .WORD   ^M<>                  ; ENTRY MASK.
0000001D'EF    17   073A  1454          JMP     RESRVD_ENTRY
```

```
                        0740  1456          .ALIGN  QUAD
                        0740  1457 ; RESERVED ENTRY POINT.                    ; ^X80000540
                   0000 0740  1458          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 0742  1459          JMP     RESRVD_ENTRY
                        0748  1460
                        0748  1461          .ALIGN  QUAD
                        0748  1462 ; RESERVED ENTRY POINT.
                   0000 0748  1463          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 074A  1464          JMP     RESRVD_ENTRY
                        0750  1465
                        0750  1466          .ALIGN  QUAD
                        0750  1467 ; RESERVED ENTRY POINT.
                   0000 0750  1468          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 0752  1469          JMP     RESRVD_ENTRY
                        0758  1470
                        0758  1471          .ALIGN  QUAD
                        0758  1472 ; RESERVED ENTRY POINT.
                   0000 0758  1473          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 075A  1474          JMP     RESRVD_ENTRY
                        0760  1475
                        0760  1476          .ALIGN  QUAD
                        0760  1477 ; RESERVED ENTRY POINT.
                   0000 0760  1478          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 0762  1479          JMP     RESRVD_ENTRY
                        0768  1480
                        0768  1481          .ALIGN  QUAD
                        0768  1482 ; RESERVED ENTRY POINT.
                   0000 0768  1483          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 076A  1484          JMP     RESRVD_ENTRY
                        0770  1485
                        0770  1486          .ALIGN  QUAD
                        0770  1487 ; RESERVED ENTRY POINT.
                   0000 0770  1488          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 0772  1489          JMP     RESRVD_ENTRY
                        0778  1490
                        0778  1491          .ALIGN  QUAD
                        0778  1492 ; RESERVED ENTRY POINT.
                   0000 0778  1493          .WORD   ^M<>                      ; ENTRY MASK.
          0000001D'EF   17 077A  1494          JMP     RESRVD_ENTRY
```

```
                        0780  1496          .ALIGN  QUAD
                        0780  1497 ; RESERVED ENTRY POINT.                    ; ^X80000580
                0000    0780  1498          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0782  1499          JMP     RESRVD_ENTRY
                        0788  1500
                        0788  1501          .ALIGN  QUAD
                        0788  1502 ; RESERVED ENTRY POINT.
                0000    0788  1503          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    078A  1504          JMP     RESRVD_ENTRY
                        0790  1505
                        0790  1506          .ALIGN  QUAD
                        0790  1507 ; RESERVED ENTRY POINT.
                0000    0790  1508          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0792  1509          JMP     RESRVD_ENTRY
                        0798  1510
                        0798  1511          .ALIGN  QUAD
                        0798  1512 ; RESERVED ENTRY POINT.
                0000    0798  1513          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    079A  1514          JMP     RESRVD_ENTRY
                        07A0  1515
                        07A0  1516          .ALIGN  QUAD
                        07A0  1517 ; RESERVED ENTRY POINT.
                0000    07A0  1518          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    07A2  1519          JMP     RESRVD_ENTRY
                        07A8  1520
                        07A8  1521          .ALIGN  QUAD
                        07A8  1522 ; RESERVED ENTRY POINT.
                0000    07A8  1523          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    07AA  1524          JMP     RESRVD_ENTRY
                        07B0  1525
                        07B0  1526          .ALIGN  QUAD
                        07B0  1527 ; RESERVED ENTRY POINT.
                0000    07B0  1528          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    07B2  1529          JMP     RESRVD_ENTRY
                        07B8  1530
                        07B8  1531          .ALIGN  QUAD
                        07B8  1532 ; RESERVED ENTRY POINT.
                0000    07B8  1533          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    07BA  1534          JMP     RESRVD_ENTRY
```

```
                          07C0  1536          .ALIGN  QUAD
                          07C0  1537 ; RESERVED ENTRY POINT.                  ; ^X80005C0
                    0000  07C0  1538          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07C2  1539          JMP     RESRVD_ENTRY
                          07C8  1540
                          07C8  1541          .ALIGN  QUAD
                          07C8  1542 ; RESERVED ENTRY POINT.
                    0000  07C8  1543          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07CA  1544          JMP     RESRVD_ENTRY
                          07D0  1545
                          07D0  1546          .ALIGN  QUAD
                          07D0  1547 ; RESERVED ENTRY POINT.
                    0000  07D0  1548          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07D2  1549          JMP     RESRVD_ENTRY
                          07D8  1550
                          07D8  1551          .ALIGN  QUAD
                          07D8  1552 ; RESERVED ENTRY POINT.
                    0000  07D8  1553          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07DA  1554          JMP     RESRVD_ENTRY
                          07E0  1555
                          07E0  1556          .ALIGN  QUAD
                          07E0  1557 ; RESERVED ENTRY POINT.
                    0000  07E0  1558          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07E2  1559          JMP     RESRVD_ENTRY
                          07E8  1560
                          07E8  1561          .ALIGN  QUAD
                          07E8  1562 ; RESERVED ENTRY POINT.
                    0000  07E8  1563          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07EA  1564          JMP     RESRVD_ENTRY
                          07F0  1565
                          07F0  1566          .ALIGN  QUAD
                          07F0  1567 ; RESERVED ENTRY POINT.
                    0000  07F0  1568          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07F2  1569          JMP     RESRVD_ENTRY
                          07F8  1570
                          07F8  1571          .ALIGN  QUAD
                          07F8  1572 ; RESERVED ENTRY POINT.
                    0000  07F8  1573          .WORD   ^M<>                  ; ENTRY MASK.
          0000001D'EF   17  07FA  1574          JMP     RESRVD_ENTRY
                          0800  1575
                          0800  1576          .ALIGN  QUAD
                 000007FF 0800  1577 DS$AQ_SSEND      == . - 1
                 00000600 0800  1578 DS$K_SSSIZE      == . - DS$AQ_SYSSRV
```

```
                          0800  1580 ;++
                          0800  1581 ; FUNCTIONAL DESCRIPTION:
                          0800  1582 ;
                          0800  1583 ;       These entry points are fixed to allow VMS device drivers
                          0800  1584 ;       a static program interface to the common utility and
                          0800  1585 ;       support routines.  Note that the absolute addresses
                          0800  1586 ;       of these entry points are defined only in $QIOVEC_DEF in
                          0800  1587 ;       DS.MLB.
                          0800  1588 ;
                          0800  1589 ;--
                          0800  1590 SYS$GL_OPRMBX::
                 00000000 0800  1591         .LONG   0                       ; ^X10800
                          0804  1592 MMG$GL_SPTBASE::
                 00000000 0804  1593         .LONG   0                       ; ^X10804
                          0808  1594 MMG$GL_P0BASE::
                 00000000' 0808 1595         .LONG   P0PT_BASE               ; ^X10808
                          080C  1596 SCH$GL_PCBVEC::
                 00000000' 080C 1597         .ADDRESS        DS$GL_PCBVEC    ; Address of PCB vector
                 00000000 0810  1598 EXE$GL_ABSTIM:: .LONG   0                       ; EXE$GL_ABSTIM, time in secs
                 00000000 0814  1599                 .LONG   0                       ; Unused
                 00000818' 0818 1600 IOC$GL_PSFL::   .LONG   IOC$GL_PSFL
                 00000818' 081C 1601 IOC$GL_PSBL::   .LONG   IOC$GL_PSFL
                          0820  1602         .ALIGN  QUAD
      00000000'9F    17  0820  1603         JMP     @#ACP$ACCESS
                          0826  1604         .ALIGN  QUAD
      00000000'9F    17  0828  1605         JMP     @#ACP$DEACCESS
                          082E  1606         .ALIGN  QUAD
      00000000'9F    17  0830  1607         JMP     @#ACP$MODIFY
                          0836  1608         .ALIGN  QUAD
      00000000'9F    17  0838  1609         JMP     @#ACP$MOUNT
                          083E  1610         .ALIGN  QUAD
      00000000'9F    17  0840  1611         JMP     @#ACP$READBLK
                          0846  1612         .ALIGN  QUAD
      00000000'9F    17  0848  1613         JMP     @#ACP$WRITEBLK
                          084E  1614         .ALIGN  QUAD
      00000000'9F    17  0850  1615         JMP     @#ERL$DEVICERR
                          0856  1616         .ALIGN  QUAD
      00000000'9F    17  0858  1617         JMP     @#ERL$DEVICTMO
                          085E  1618         .ALIGN  QUAD
      00000000'9F    17  0860  1619         JMP     @#EXE$ABORTIO
                          0866  1620         .ALIGN  QUAD
      00000000'9F    17  0868  1621         JMP     @#EXE$IOFORK
                          086E  1622         .ALIGN  QUAD
      00000000'9F    17  0870  1623         JMP     @#EXE$ONEPARM
                          0876  1624         .ALIGN  QUAD
      00000000'9F    17  0878  1625         JMP     @#EXE$PWRTIMCHK
                          087E  1626         .ALIGN  QUAD
      00000000'9F    17  0880  1627         JMP     @#EXE$SENSEMODE
                          0886  1628         .ALIGN  QUAD
      00000000'9F    17  0888  1629         JMP     @#EXE$SETCHAR
                          088E  1630         .ALIGN  QUAD
      00000000'9F    17  0890  1631         JMP     @#EXE$SETMODE
                          0896  1632         .ALIGN  QUAD
      00000000'9F    17  0898  1633         JMP     @#EXE$SNDEVMSG
                          089E  1634         .ALIGN  QUAD
      00000000'9F    17  08A0  1635         JMP     @#EXE$ZEROPARM
                          08A6  1636         .ALIGN  QUAD
```

```
                00000000'9F    17  08A8  1637          JMP     @#IOC$APPLYECC
                               08AE  1638          .ALIGN  QUAD
                00000000'9F    17  08B0  1639          JMP     @#IOC$CANCELIO
                               08B6  1640          .ALIGN  QUAD
                00000000'9F    17  08B8  1641          JMP     @#IOC$DIAGBUFILL
                               08BE  1642          .ALIGN  QUAD
                00000000'9F    17  08C0  1643          JMP     @#IOC$LOADMBAMAP
                               08C6  1644          .ALIGN  QUAD
                00000000'9F    17  08C8  1645          JMP     @#IOC$LOADUBAMAP
                               08CE  1646          .ALIGN  QUAD
                00000000'9F    17  08D0  1647          JMP     @#IOC$MOVTOUSER
                               08D6  1648          .ALIGN  QUAD
                00000000'9F    17  08D8  1649          JMP     @#IOC$PURGDATAP
                               08DE  1650          .ALIGN  QUAD
                00000000'9F    17  08E0  1651          JMP     @#IOC$RELCHAN
                               08E6  1652          .ALIGN  QUAD
                00000000'9F    17  08E8  1653          JMP     @#IOC$RELDATAP
                               08EE  1654          .ALIGN  QUAD
                00000000'9F    17  08F0  1655          JMP     @#IOC$RELMAPREG
                               08F6  1656          .ALIGN  QUAD
                00000000'9F    17  08F8  1657          JMP     @#IOC$RELSCHAN
                               08FE  1658          .ALIGN  QUAD
                00000000'9F    17  0900  1659          JMP     @#IOC$REQCOM
                               0906  1660          .ALIGN  QUAD
                00000000'9F    17  0908  1661          JMP     @#IOC$REQDATAP
                               090E  1662          .ALIGN  QUAD
                00000000'9F    17  0910  1663          JMP     @#IOC$REQMAPREG
                               0916  1664          .ALIGN  QUAD
                00000000'9F    17  0918  1665          JMP     @#IOC$REQPCHANH
                               091E  1666          .ALIGN  QUAD
                00000000'9F    17  0920  1667          JMP     @#IOC$REQPCHANL
                               0926  1668          .ALIGN  QUAD
                00000000'9F    17  0928  1669          JMP     @#IOC$REQSCHANL
                               092E  1670          .ALIGN  QUAD
                00000000'9F    17  0930  1671  IOC$RTN::JMP     @#IOC$RETURN            ; Defined external for QIO
                               0936  1672          .ALIGN  QUAD
                00000000'9F    17  0938  1673          JMP     @#IOC$SENSEDISK
                               093E  1674          .ALIGN  QUAD
                00000000'9F    17  0940  1675          JMP     @#IOC$UPDATRANSP
                               0946  1676          .ALIGN  QUAD
                00000000'9F    17  0948  1677          JMP     @#IOC$WFIKPCH
                               094E  1678          .ALIGN  QUAD
                00000000'9F    17  0950  1679          JMP     @#IOC$WFIRLCH
                               0956  1680          .ALIGN  QUAD
                00000000'9F    17  0958  1681          JMP     @#MT$CHECK_ACCESS
                               095E  1682          .ALIGN  QUAD
                00000000'9F    17  0960  1683          JMP     @#EXE$ALLOCIRP
                               0966  1684          .ALIGN  QUAD
                00000000'9F    17  0968  1685          JMP     @#EXE$ALONONPAGED
                               096E  1686          .ALIGN  QUAD
                00000000'9F    17  0970  1687          JMP     @#EXE$DEANONPAGED
                               0976  1688          .ALIGN  QUAD
                00000000'9F    17  0978  1689          JMP     @#EXE$FINISHIOC
                               097E  1690          .ALIGN  QUAD
                00000000'9F    17  0980  1691          JMP     @#EXE$FORK
                               0986  1692          .ALIGN  QUAD
                00000000'9F    17  0988  1693          JMP     @#EXE$MODIFYLOCKR
```

```
                        098E  1694          .ALIGN  QUAD
00000000'9F    17       0990  1695          JMP     @#EXE$QIODRVPKT
                        0996  1696          .ALIGN  QUAD
00000000'9F    17       0998  1697          JMP     @#EXE$WRITELOCK
                        099E  1698          .ALIGN  QUAD
00000000'9F    17       09A0  1699          JMP     @#SCH$POSTEF
                        09A6  1700          .ALIGN  QUAD
00000000'9F    17       09A8  1701          JMP     @#SCH$QAST
                        09AE  1702          .ALIGN  QUAD
00000000'9F    17       09B0  1703          JMP     @#EXE$WRITECHK
                        09B6  1704          .ALIGN  QUAD
00000000'9F    17       09B8  1705          JMP     @#EXE$READLOCKR
                        09BE  1706          .ALIGN  QUAD
00000000'9F    17       09C0  1707          JMP     @#EXE$WRITELOCKR
                        C9C6  1708          .ALIGN QUAD
00000000'9F    17       09C8  1709          JMP     @#IOC$INITIATE
                        09CE  1710          .ALIGN  QUAD
00000000'9F    17       09D0  1711          JMP     @#EXE$INSERTIRP
                        09D6  1712          .ALIGN QUAD
00000000'9F    17       09D8  1713          JMP     @#EXE$QIORETURN
                        09DE  1714          .ALIGN   QUAD
00000000'9F    17       09E0  1715          JMP     @#IOC$REQDATAPNW
                        09E6  1716          .ALIGN  QUAD
00000000'9F    17       09E8  1717          JMP     @#COM$POST
                        09EE  1718          .ALIGN  QUAD
00000000'9F    17       09F0  1719          JMP     @#IOC$ALOUBAMAP
                        09F6  1720          .ALIGN  QUAD
00000000'9F    17       09F8  1721          JMP     @#IOC$ALTUBAMAP
                        09FE  1722          .ALIGN  QUAD
00000000'9F    17       0A00  1723          JMP     @#ERL$RELEASEMB
                        0A06  1724          .ALIGN  QUAD
00000000'9F    17       0A08  1725          JMP     @#IOC$ALOURAMAPN
                        0A0E  1726          .ALIGN  QUAD
00000000'9F    17       0A10  1727          JMP     @#EXE$ALLOCBUF
                        0A16  1728          .ALIGN  QUAD
00000000'9F    17       0A18  1729          JMP     @#EXE$BUFQUOPRC
                        0A1E  1730          .ALIGN  QUAD
00000000'9F    17       0A20  1731          JMP     @#EXE$BUFFRQUOTA
                        0A26  1732          .ALIGN  QUAD
00000000'9F    17       0A28  1733          JMP     @#EXE$FINISHIO
                        0A2E  1734          .ALIGN  QUAD
00000000'9F    17       0A30  1735          JMP     @#EXE$READ
                        0A36  1736          .ALIGN  QUAD
00000000'9F    17       0A38  1737          JMP     @#EXE$READCHK
                        0A3E  1738          .ALIGN  QUAD
00000000'9F    17       0A40  1739          JMP     @#EXE$READCHKR
                        0A46  1740          .ALIGN  QUAD
00000000'9F    17       0A48  1741          JMP     @#EXE$WRITE
                        0A4E  1742          .ALIGN  QUAD
00000000'9F    17       0A50  1743          JMP     @#EXE$WRITECHKR
                        0A56  1744          .ALIGN  QUAD
00000000'9F    17       0A58  1745          JMP     @#EXE$MODIFY
                        0A5E  1746          .ALIGN  QUAD
00000000'9F    17       0A60  1747          JMP     @#EXE$MODIFYLOCK
                        0A66  1748          .ALIGN  QUAD
00000000'9F    17       0A68  1749          JMP     @#EXE$CARRIAGE
                        0A6E  1750          .ALIGN  QUAD
```

```
                00000000'9F   17  0A70  1751          JMP     @#IOC$REQSCHANH
                                  0A76  1752          .ALIGN QUAD
                00000000'9F   17  0A78  1753          JMP     @#IOC$LOADUBAMAPA
                                  0A7E  1754          .ALIGN QUAD
                00000000'9F   17  0A80  1755          JMP     @#IOC$GETBYTE
                                  0A86  1756          .ALIGN QUAD
                00000000'9F   17  0A88  1757          JMP     @#IOC$PUTBYTE
                                  0A8E  1758          .ALIGN QUAD
                00000000'9F   17  0A90  1759          JMP     @#IOC$INITBUFWIND
                                  0A96  1760          .ALIGN QUAD
                00000000'9F   17  0A98  1761          JMP     @#IOC$MOVFRUSER2
                                  0A9E  1762          .ALIGN QUAD
                00000000'9F   17  0AA0  1763          JMP     @#IOC$MOVFRUSER1
                                  0AA6  1764          .ALIGN QUAD
                00000000'9F   17  0AA8  1765          JMP     @#IOC$MOVTOUSER2
                                  0AAE  1766          .ALIGN QUAD
                00000000'9F   17  0AB0  1767          JMP     @#IOC$MOVTOUSER1
                                  0AB6  1768          .ALIGN  QUAD
                00000000'9F   17  0AB8  1769          JMP     @#SCH$LOCKW                     ; Lock 'MUTEX'' for write access
                                  0ABE  1770          .ALIGN  QUAD
                00000000'9F   17  0AC0  1771          JMP     @#SCH$UNLOCK                    ; Unlock 'MUTEX''
                                  0AC6  1772          .ALIGN  QUAD
                00000000'9F   17  0AC8  1773          JMP     @#EXE$ALTQUEPKT
                                  0ACE  1774          .ALIGN  QUAD
                00000000'9F   17  0AD0  1775          JMP     @#COM$SETATTNAST
                                  0AD6  1776          .ALIGN  QUAD
                00000000'9F   17  0AD8  1777          JMP     @#COM$DELATTNAST
                                  0ADE  1778          .ALIGN  QUAD
                00000000'9F   17  0AE0  1779          JMP     @#COM$FLUSHATTNS
                                  0AE6  1780          .ALIGN  QUAD
                00000000'9F   17  0AE8  1781          JMP     @#COM$DRVDEALMEM
                                  0AEE  1782          .ALIGN  QUAD
                00000000'9F   17  0AF0  1783          JMP     @#EXE$INSIOQ
                                  0AF6  1784          .ALIGN  QUAD
                00000000'9F   17  0AF8  1785          JMP     @#IOC$PTETOPFN
                                  0AFE  1786          .ALIGN  QUAD
                                  0B00  1787 EXE$GQ_SYSTIME::
            00000000 00000000     0B00  1788          .QUAD   0                       ; **DATA** System time quadword
                                  0B08  1789          .ALIGN  QUAD
                00000000'9F   17  0B08  1790          JMP     @#IOC$MOVFRUSER
                                  0B0E  1791          .ALIGN  QUAD
                00000000'9F   17  0B10  1792          JMP     @#IOC$ALLOSPT                   ;                               [23]
                                  0B16  1793          .ALIGN  QUAD                    ;                               [23]
                00000000'9F   17  0B18  1794          JMP     @#EXE$MAXACMODE         ;                               [23]
                                  0B1E  1795          .ALIGN  QUAD                    ;                               [23]
                00000000'9F   17  0B20  1796          JMP     @#EXE$READLOCK          ;                               [23]
                                  0B26  1797          .ALIGN  QUAD                    ;                               [23]
                                  0B28  1798
```

```
                        0828  1800 .sbttl DS$SRVDISP_TABLE - Dispatch table for VDS services
                    00000034  1801 .psect DATA, SHR, NOEXE, NOWRT, QUAD
                        0034  1802
                        0034  1803 ; This is the dispatch table for all future additional VDS services.    [35]
                        0034  1804
                        0034  1805                 .ALIGN QUAD                ;                           [35]
                        0038  1806 DS$SRVDISP_TABLE::                        ;                           [35]
                        0038  1807
                  0000'     0038  1808         .VECTOR DSX$MOUNT              ;                           [35]
      00000002'EF   17      003A  1809         JMP     DSX$MOUNT+2            ;                           [35]
                        0040  1610
                  0000'     0040  1811         .VECTOR DSX$DISMOUNT           ;                           [35]
      00000002'EF   17      0042  1812         JMP     DSX$DISMOUNT+2         ;                           [35]
                        0048  1813
                        C048  1814 DS$SRVDISP_TABLE_END:                      ;                           [35]
                        0048  1815
```

```
                     0048  1817 .SBTTL DSX$SERVICE_DISPATCHER - Service Dispatch Routine
                 00000000  1818 .psect code, shr, exe, nowrt, byte
                     0000  1819
                     0000  1820 ;++                                                                          [35]
                     0000  1821 ; Functional Description:                                                  [35]
                     0000  1822 ;                                                                          [35]
                     0000  1823 ;       Calls the proper service via the service dispatch table.          [35]
                     0000  1824 ;                                                                          [35]
                     0000  1825 ; Inputs:                                                                  [35]
                     0000  1826 ;                                                                          [35]
                     0000  1827 ;       .................................                                  [35]
                     0000  1828 ;       :       No. of args.            :<=AP                              [35]
                     0000  1829 ;       :.............................:                                     [35]
                     0000  1830 ;       :Code indicating type of service:                                  [35]
                     0000  1831 ;       :.............................:                                     [35]
                     0000  1832 ;       :       Service routine arg1    :                                  [35]
                     0000  1833 ;       :.............................:                                     [35]
                     0000  1834 ;       /                               /                                  [35]
                     0000  1835 ;       /                               /                                  [35]
                     0000  1836 ;       :.............................:                                     [35]
                     0000  1837 ;       :       Service routine argN    :                                  [35]
                     0000  1838 ;       :.............................:                                     [35]
                     0000  1839 ;                                                                          [35]
                     0000  1840 ;                                                                          [35]
                     0000  1841 ; Implicit Inputs:                                                        [35]
                     0000  1842 ;                                                                          [35]
                     0000  1843 ; Outputs:                                                                [35]
                     0000  1844 ;                                                                          [35]
                     0000  1845 ; Implicit Outputs:                                                       [35]
                     0000  1846 ;                                                                          [35]
                     0000  1847 ; Side Effects:                                                           [35]
                     0000  1848 ;                                                                          [35]
                     0000  1849 ; Return Status:                                                          [35]
                     0000  1850 ;                                                                          [35]
                     0000  1851 ;--                                                                       [35]
                     0000  1852
                0000 0000  1853 .ENTRY DSX$SERVICE_DISPATCHER, ^M<>     ;                                  [35]
                     0002  1854
 7E   04 AC    08   C5  0002  1855         MULL3   #8,4(AP),-(SP)          ; Calc. offset into table.     [35]
7E   00000038'EF   9E  0007  1856         MOVAB   DS$SRVDISP_TABLE,-(SP)  ; Get dispatch table address.  [35]
     04 AE    6E   C0  000E  1857         ADDL    (SP),4(SP)              ; Add offset.                  [35]
 04 AC    6C   01   C3  0012  1858         SUBL3   #1,(AP),4(AP)           ; Create true arg. list for    [35]
                        0017  1859                                         ;   service routine by getting [35]
                        0017  1860                                         ;   rid of service code arg.    [35]
   04 BE    04 AC    FA  0017  1861         CALLG   4(AP),@4(SP)            ; Call the proper service.     [35]
                   04   001C  1862         RET                             ; Return.                      [35]
                        001D  1863
```

```
                     001D  1865          .SBTTL  Resrvd_Entry Routine - For entry errors
                00000001D  1866          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     001D  1867 ;++
                     001D  1868 ; FUNCTIONAL DESCRIPTION:
                     001D  1869 ;
                     001D  1870 ;
                     001D  1871 ; CALLING SEQUENCE:
                     001D  1872 ;
                     001D  1873 ;       NONE
                     001D  1874 ;
                     001D  1875 ; INPUT PARAMETERS:
                     001D  1876 ;
                     001D  1877 ;       NONE
                     001D  1878 ;
                     C01D  1879 ; IMPLICIT INPUTS:
                     001D  1880 ;
                     001D  1881 ;       NONE
                     001D  1882 ;
                     001D  1883 ; OUTPUT PARAMETERS:
                     001D  1884 ;
                     001D  1885 ;       NONE
                     001D  1886 ;
                     001D  1887 ; IMPLICIT OUTPUTS:
                     001D  1888 ;
                     001D  1889 ;       NONE
                     001D  1890 ;
                     001D  1891 ; COMPLETION CODES:
                     001D  1892 ;
                     001D  1893 ;       NONE
                     001D  1894 ;
                     001D  1895 ; SIDE EFFECTS:
                     001D  1896 ;
                     001D  1897 ;       NONE
                     001D  1898 ;
                     001D  1899 ;--
```

```
                          001D  1901 RESRVD_ENTRY:
                          001D  1902         ERRSUP_S        MSGADR=L^T_RESRVD       ;                                [25]
                          0032  1903
                          0032  1904 FAKE_JUMP:                                     ;                                [27]
            00000000'EF    16 0032  1905         JSB     KB_CHECK                    ; Check keyboard status          [25]
                           04 0038  1906         RET
                          0039  1907
                          0039  1908         .END
```

```
$ER                             = 00000002    D
$MODULE                           00000000 R  D  01
ACP$ACCESS                        ********    X  02
ACP$DEACCESS                      ********    X  02
ACP$MODIFY                        ********    X  02
ACP$MOUNT                         ********    X  02
ACP$READBLK                       ********    X  02
ACP$WRITEBLK                      ********    X  02
APT                               ********    X  02
BOOT                              ********    X  02
COM$DELATTNAST                    ********    X  02
COM$DRVDEALMEM                    ********    X  02
COM$FLUSHATTNS                    ********    X  02
COM$POST                          ********    X  02
COM$SETATTNAST                    ********    X  02
CR                              = 0000000D    D
DS$ABORT                          00000020 RG D  02
DS$ABORTWAIT                      00000070 RG D  02
DS$APT_ENTRY                      00000004 R  D  02
DS$AQ_SSEND                     = 000007FF RG D  02
DS$AQ_SYSSRV                      00000200 RG D  02
DS$ASKADR                         00000090 RG D  02
DS$ASKDATA                        00000080 RG D  02
DS$ASKLGCL                        00000098 RG D  02
DS$ASKSTR                         000000A0 RG D  02
DS$ASKVLD                         00000088 RG D  02
DS$ATTACH                         000001A8 RG D  02
DS$BGNSUB                         00000030 RG D  02
DS$BRANCH                         000000A8 RG D  02
DS$BREAK                          00000058 RG D  02
DS$CANWAIT                        00000070 RG D  02
DS$CHANNEL                        00000180 RG D  02
DS$CKLOOP                         00000040 RG D  02
DS$CLRVEC                         00000168 RG D  02
DS$CNTRLC                         00000078 RG D  02
DS$CVTREG                         000000B0 RG D  02
DS$DOSUMMARY                      00000028 RG D  02
DS$ENDPASS                        00000010 RG D  02
DS$ENDSUB                         00000038 RG D  02
DS$ENTRY                          00000000 R  D  02
DS$ERRDEV                         000000C8 RG D  02
DS$ERRHARD                        000000D0 RG D  02
DS$ERRPREP                        00000108 RG D  02
DS$ERRSOFT                        000000D8 RG D  02
DS$ERRSYS                         000000C0 RG D  02
DS$ESCAPE                         00000050 RG D  02
DS$FREEDBGSYM                     000001B8 RG D  02
DS$GETADDRESS                     00000090 RG D  02
DS$GETBUF                         00000120 RG D  02
DS$GETDATA                        00000080 RG D  02
DS$GETLOGICAL                     00000098 RG D  02
DS$GETSTRING                      000000A0 RG D  02
DS$GETTERM                        000001C8 RG D  02
DS$GETVIELD                       00000088 RG D  02
DS$GL_PCBVEC                      ********    X  02
DS$GPHARD                         00000018 RG D  02
DS$HELP                           000001B0 RG D  02
DS$INITSCB                        00000170 RG D  02
DS$INLOOP                         00000048 RG D  02
DS$K_SSSIZE                     = 00000600  G D
DS$LOAD                           00000198 RG D  02
DS$LOADPCS                        000001C0 RG D  02
DS$MAPDBGBLOCK                    00000118 RG D  02
DS$MMOFF                          00000158 RG D  02
DS$MMON                           00000150 RG D  02
DS$PARSE                          000000B8 RG D  02
DS$PRINTB                         000000E0 RG D  02
DS$PRINTF                         000000F0 RG D  02
DS$PRINTS                         000000F8 RG D  02
DS$PRINTSIG                       00000100 RG D  02
DS$PRINTX                         000000E8 RG D  02
DS$PRC3E                          000001A0 RG D  02
DS$RELBUF                         00000128 RG D  02
DS$SERVICE_DISPATCHER             000001D0 RG D  02
DS$SETIPL                         00000178 RG D  02
DS$SETMAP                         00000188 RG D  02
DS$SETPRIEXV                      00000110 RG D  02
DS$SETVEC                         00000160 RG D  02
DS$SHOCHAN                        00000190 RG D  02
DS$SHOWCHAN                       00000190 RG D  02
DS$SRVDISP_TABLE                  00000038 RG D  01
DS$SRVDISP_TABLE_END              00000048 R  D  01
DS$SUMMARY                        00000028 RG D  02
DS$WAITMS                         00000060 RG D  02
DS$WAITUS                         00000068 RG D  02
DSX$ABORT                         ********    X  02
DSX$ATTACH                        ********    X  02
DSX$BGNSUB                        ********    X  02
DSX$BRANCH                        ********    X  02
DSX$CANWAIT                       ********    X  02
DSX$CHANNEL                       ********    X  02
DSX$CKLOOP                        ********    X  02
DSX$CLRVEC                        ********    X  02
DSX$CNTRLC                        ********    X  02
DSX$CVTREG                        ********    X  02
DSX$DISMOUNT                      ********    X  01
DSX$DOSUMMARY                     ********    X  02
DSX$ENDPASS                       ********    X  02
DSX$ENDSUB                        ********    X  02
DSX$ERRDEV                        ********    X  02
DSX$ERRHARD                       ********    X  02
DSX$ERRPREP                       ********    X  02
DSX$ERRSOFT                       ********    X  02
DSX$ERRSYS                        ********    X  02
DSX$ESCAPE                        ********    X  02
DSX$FREEDBGSYM                    ********    X  02
DSX$GETADDRESS                    ********    X  02
DSX$GETBUF                        ********    X  02
DSX$GETDATA                       ********    X  02
DSX$GETLOGICAL                    ********    X  02
DSX$GETSTRING                     ********    X  02
DSX$GETTERM                       ********    X  02
DSX$GETVIELD                      ********    X  02
DSX$GPHARD                        ********    X  02
```

DSX$HELP	********	X	02	EXE$HIBER	********	X	02
DSX$INITSCB	********	X	02	EXE$INSERTIRP	********	X	02
DSX$INLOOP	********	X	02	EXE$INSIOQ	********	X	02
DSX$LOAD	********	X	02	EXE$IOFORK	********	X	02
DSX$LOADPCS	********	X	02	EXE$MAXACMODE	********	X	02
DSX$MAPDBGBLOCK	********	X	02	EXE$MODIFY	********	X	02
DSX$MMOFF	********	X	02	EXE$MODIFYLOCK	********	X	02
DSX$MMON	********	X	02	EXE$MODIFYLOCKR	********	X	02
DSX$MOUNT	********	X	01	EXE$NUMTIM	********	X	02
DSX$PARSE	********	X	02	EXE$ONEPARM	********	X	02
DSX$PRINTB	********	X	02	EXE$PWRTIMCHK	********	X	02
DSX$PRINTF	********	X	02	EXE$QIO	********	X	02
DSX$PRINTS	********	X	02	EXE$QIO2	= 00000204	R	D	02
DSX$PRINTSIG	********	X	02	EXE$QIODRVPKT	********	X	02
DSX$PRINTX	********	X	02	EXE$QIORETURN	********	X	02
DSX$PROBE	********	X	02	EXE$READ	********	X	02
DSX$RELBUF	********	X	02	EXE$READCHK	********	X	02
DSX$SERVICE_DISPATCHER	00000000	RG	D	03	EXE$READCHKR	********	X	02
DSX$SETIPL	********	X	02	EXE$READEF	********	X	02
DSX$SETMAP	********	X	02	EXE$READLOCK	********	X	02
DSX$SETPRIEXV	********	X	02	EXE$READLOCKR	********	X	02
DSX$SETVEC	********	X	02	EXE$SENSEMODE	********	X	02
DSX$SHOWCHAN	********	X	02	EXE$SETAST	********	X	02
DSX$WAITMS	********	X	02	EXE$SETCHAR	********	X	02
DSX$WAITUS	********	X	02	EXE$SETEF	********	X	02
DS_ERRSUP	********	X	03	EXE$SETIMR	********	X	02
ERL$DEVICERR	********	X	02	EXE$SETMODE	********	X	02
ERL$DEVICTMO	********	X	02	EXE$SETPRT	********	X	02
ERL$RELEASEMB	********	X	02	EXE$SNDEVMSG	********	X	02
EXE$ABORTIO	********	X	02	EXE$UNWIND	********	X	02
EXE$ALLOC	********	X	02	EXE$WAITFR	********	X	02
EXE$ALLOCBUF	********	X	02	EXE$WAKE	********	X	02
EXE$ALLOCIRP	********	X	02	EXE$WFLAND	********	X	02
EXE$ALONONPAGED	********	X	02	EXE$WFLOR	********	X	02
EXE$ALTQUEPKT	********	X	02	EXE$WRITE	********	X	02
EXE$ASCTIM	********	X	02	EXE$WRITECHK	********	X	02
EXE$ASSIGN	********	X	02	EXE$WRITECHKR	********	X	02
EXE$BINTIM	********	X	02	EXE$WRITELOCK	********	X	02
EXE$BUFFRQUOTA	********	X	02	EXE$WRITELOCKR	********	X	02
EXE$BUFQUOPRC	********	X	02	EXE$ZEROPARM	********	X	02
EXE$CANCEL	********	X	02	FAKE_JUMP	00000032	R	D	03
EXE$CANTIM	********	X	02	IOC$ALLOSPT	********	X	02
EXE$CARRIAGE	********	X	02	IOC$ALOUBAMAP	********	X	02
EXE$CLREF	********	X	02	IOC$ALOUBAMAPN	********	X	02
EXE$CNTREG	********	X	02	IOC$ALTUBAMAP	********	X	02
EXE$DASSGN	********	X	02	IOC$APPLYECC	********	X	02
EXE$DEANONPAGED	********	X	02	IOC$CANCELIO	********	X	02
EXE$EXPREG	********	X	02	IOC$DIAGBUFILL	********	X	02
EXE$FAO	********	X	02	IOC$GETBYTE	********	X	02
EXE$FAOL	********	X	02	IOC$GL_PSBL	0000081C	RG	D	02
EXE$FINISHIO	********	X	02	IOC$GL_PSFL	00000818	RG	D	02
EXE$FINISHIOC	********	X	02	IOC$INITBUFWIND	********	X	02
EXE$FORK	********	X	02	IOC$INITIATE	********	X	02
EXE$GETTIM	********	X	02	IOC$LOADMBAMAP	********	X	02
EXE$GL_ABSTIM	00000810	RG	D	02	IOC$LOADUBAMAP	********	X	02
EXE$GQ_SYSTIME	00000B00	RG	D	02	IOC$LOADUBAMAPA	********	X	02
EXE$GTCHAN	********	X	02	IOC$MOVFRUSER	********	X	02

IOC$MOVFRUSER1	********	X	02	SYS$CONNECT	000005C0	RG	D	02
IOC$MOVFRUSER2	********	X	02	SYS$DALLOC	000002D8	RG	D	02
IOC$MOVTOUSER	********	X	02	SYS$DASSGN	000002E0	RG	D	02
IOC$MOVTOUSER1	********	X	02	SYS$DISCONNECT	000005D0	RG	D	02
IOC$MOVTOUSER2	********	X	02	SYS$EXIT	00000340	RG	D	02
IOC$PTETOPFN	********	X	02	SYS$EXPREG	00000348	RG	D	02
IOC$PURGDATAP	********	X	02	SYS$FAO	00000350	RG	D	02
IOC$PUTBYTE	********	X	02	SYS$FAOL	00000358	RG	D	02
IOC$RELCHAN	********	X	02	SYS$GET	00000580	RG	D	02
IOC$RELDATAP	********	X	02	SYS$GETCHN	000004C8	RG	D	02
IOC$RELMAPREG	********	X	02	SYS$GETTIM	00000378	RG	D	02
IOC$RELSCHAN	********	X	02	SYS$GL_OPRMBX	00000800	RG	D	02
IOC$REQCOM	********	X	02	SYS$GTCHAN	00000380	RG	D	02
IOC$REQDATAP	********	X	02	SYS$HIBER	00000388	RG	D	02
IOC$REQDATAPNW	********	X	02	SYS$LKWSET	000003A0	RG	D	02
IOC$REQMAPREG	********	X	02	SYS$NUMTIM	000003B8	RG	D	02
IOC$REQPCHANH	********	X	02	SYS$OPEN	00000608	RG	D	02
IOC$REQPCHANL	********	X	02	SYS$QIO	000003C8	RG	D	02
IOC$REQSCHANH	********	X	02	SYS$QIOW	00000200	RG	D	02
IOC$REQSCHANL	********	X	02	SYS$READ	00000590	RG	D	02
IOC$RETURN	********	X	02	SYS$READEF	000003D0	RG	D	02
IOC$RTN	00000930	RG	D 02	SYS$SETAST	000003F8	RG	D	02
IOC$SENSEDISK	********	X	02	SYS$SETEF	00000400	RG	D	02
IOC$UPDATRANSP	********	X	02	SYS$SETIMR	00000420	RG	D	02
IOC$WFIKPCH	********	X	02	SYS$SETPRI	00000428	RG	D	02
IOC$WFIRLCH	********	X	02	SYS$SETPRT	00000430	RG	D	02
KB_CHECK	********	X	03	SYS$SETRWM	00000438	RG	D	02
LEAP_APT	000001F4	R	D 02	SYS$TRNLOG	00000458	RG	D	02
LEAP_BOOT	000001EC	R	D 02	SYS$ULKPAG	00000460	RG	D	02
LF	= 0000000A		D	SYS$ULWSET	00000468	RG	D	02
MMG$GL_POBASE	00000808	RG	D 02	SYS$UNWIND	00000470	RG	D	02
MMG$GL_SPTBASE	00000804	RG	D 02	SYS$WAITFR	00000478	RG	D	02
MT$CHECK_ACCESS	********	X	02	SYS$WAKE	00000480	RG	D	02
POPT_BASE	********	X	02	SYS$WFLAND	00000488	RG	D	02
RESRVD	000001FA	R	D 02	SYS$WFLOR	00000490	RG	D	02
RESRVD_ENTRY	0000001D	R	D 03	T_RESRVD	00000006	R	D	01
RMS$CLOSE	********	X	02					
RMS$CONNECT	********	X	02					
RMS$DISCONNECT	********	X	02					
RMS$GET	********	X	02					
RMS$OPEN	********	X	02					
RMS$READ	********	X	02					
SCH$GL_PCBVEC	0000080C	RG	D 02					
SCH$LOCKW	********	X	02					
SCH$POSTEF	********	X	02					
SCH$QAST	********	X	02					
SCH$UNLOCK	********	X	02					
SYS$ALLOC	00000238	RG	D 02					
SYS$ASCTIM	00000248	RG	D 02					
SYS$ASSIGN	00000250	RG	D 02					
SYS$BINTIM	00000258	RG	D 02					
SYS$CALL_HANDL	00000210	RG	D 02					
SYS$CANCEL	00000260	RG	D 02					
SYS$CANTIM	00000268	RG	D 02					
SYS$CLOSE	000005B8	RG	D 02					
SYS$CLREF	00000298	RG	D 02					
SYS$CNTREG	000002A0	RG	D 02					

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DATA	00000048 (72.)	01 (1.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC QUAD
$BASE	00000B28 (2856.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC QUAD
CODE	00000039 (57.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	1902 (1)	#-1902 (1)
$MODULE	00000000-R	201 (1)	1902 (1)
ACP$ACCESS	00000000-XR		1603 (1)
ACP$DEACCESS	00000000-XR		1605 (1)
ACP$MODIFY	00000000-XR		1607 (1)
ACP$MOUNT	00000000-XR		1609 (1)
ACP$READBLK	00000000-XR		1611 (1)
ACP$WRITEBLK	00000000-XR		1613 (1)
APT	00000000-XR		602 (1)
BOOT	00000000-XR		596 (1)
COM$DELATTNAST	00000000-XR		1777 (1)
COM$DRVDEALMEM	00000000-XR		1781 (1)
COM$FLUSHATTNS	00000000-XR		1779 (1)
COM$POST	00000000-XR		1717 (1)
COM$SETATTNAST	00000000-XR		1775 (1)
CR	=0000000D	191 (1)	204 (1)
DS$ABORT	00000020-R	279 (1)	
DS$ABORTWAIT	00000070-R	338 (1)	
DS$APT_ENTRY	00000004-R	254 (1)	
DS$AQ_SSEND	=000007FF-R	1577 (1)	
DS$AQ_SYSSRV	00000200-R	610 (1)	1578 (1)
DS$ASKADR	00000090-R	363 (1)	
DS$ASKDATA	00000080-R	351 (1)	
DS$ASKLGCL	00000098-R	369 (1)	
DS$ASKSTR	000000A0-R	375 (1)	
DS$ASKVLD	00000088-R	357 (1)	
DS$ATTACH	000001A8-R	552 (1)	
DS$BGNSUB	00000030-R	296 (1)	
DS$BRANCH	000000A8-R	382 (1)	
DS$BREAK	00000058-R	321 (1)	
DS$CANWAIT	00000070-R	337 (1)	
DS$CHANNEL	00000180-R	526 (1)	
DS$CKLOOP	00000040-R	306 (1)	
DS$CLRVEC	00000168-R	511 (1)	
DS$CNTRLC	00000078-R	343 (1)	
DS$CVTREG	000000B0-R	389 (1)	
DS$DOSUMMARY	00000028-R	288 (1)	
DS$ENDPASS	00000010-R	266 (1)	
DS$ENDSUB	00000038-R	301 (1)	
DS$ENTRY	00000000-R	250 (1)	
DS$ERRDEV	000000C8-R	404 (1)	
DS$ERRHARD	000000D0-R	409 (1)	
DS$ERRPREP	00000108-R	444 (1)	
DS$ERRSOFT	000000D8-R	414 (1)	
DS$ERRSYS	000000C0-R	399 (1)	
DS$ESCAPE	00000050-R	316 (1)	
DS$FREEDBGSYM	000001B8-R	561 (1)	
DS$GETADDRESS	00000090-R	364 (1)	
DS$GETBUF	00000120-R	463 (1)	
DS$GETDATA	00000080-R	352 (1)	

```
DS$GETLOGICAL                   00000098-R    370     (1)
DS$GETSTRING                    000000A0-R    376     (1)
DS$GETTERM                      000001C8-R    571     (1)
DS$GETVIELD                     00000088-R    358     (1)
DS$GL_PCBVEC                    00000000-XR                   1597    (1)
DS$GPHARD                       00000018-R    271     (1)
DS$HELP                         000001B0-R    556     (1)
DS$INITSCB                      00000170-R    516     (1)
DS$INLOOP                       00000048-R    311     (1)
DS$K_SSSIZE                    =00000600      1578    (1)
DS$LOAD                         00000198-R    542     (1)
DS$LOADPCS                      000001C0-R    566     (1)
DS$MAPDBGBLOCK                  00000118-R    454     (1)
DS$MMOFF                        00000158-R    498     (1)
DS$MMON                         00000150-R    493     (1)
DS$PARSE                        00000088-R    394     (1)
DS$PRINTB                       000000E0-R    419     (1)
DS$PRINTF                       000000F0-R    429     (1)
DS$PRINTS                       000000F8-R    434     (1)
DS$PRINTSIG                     00000100-R    439     (1)
DS$PRINTX                       000000E8-R    424     (1)
DS$PROBE                        000001A0-R    547     (1)
DS$RELBUF                       00000128-R    468     (1)
DS$SERVICE_DISPATCHER           000001D0-R    576     (1)
DS$SETIPL                       00000178-R    521     (1)
DS$SETMAP                       00000188-R    531     (1)
DS$SETPRIEXV                    00000110-R    449     (1)
DS$SETVEC                       00000160-R    506     (1)
DS$SHOCHAN                      00000190-R    536     (1)
DS$SHOWCHAN                     00000190-R    537     (1)
DS$SRVDISP_TABLE                00000038-R    1806    (1)     1856    (1)
DS$SRVDISP_TABLE_END            00000048-R    1814    (1)
DS$SUMMARY                      00000028-R    287     (1)
DS$WAITMS                       00000060-R    327     (1)
DS$WAITUS                       00000068-R    332     (1)
DSX$ABORT                       00000000-XR                   281     (1)
DSX$ATTACH                      00000000-XR                   554     (1)
DSX$BGNSUB                      00000000-XR                   298     (1)
DSX$BRANCH                      00000000-XR                   384     (1)
DSX$CANWAIT                     00000000-XR                   340     (1)
DSX$CHANNEL                     00000000-XR                   528     (1)
DSX$CKLOOP                      00000000-XR                   308     (1)
DSX$CLRVEC                      00000000-XR                   513     (1)
DSX$CNTRLC                      00000000-XR                   345     (1)
DSX$CVTREG                      00000000-XR                   391     (1)
DSX$DISMOUNT                    00000000-XR                   1812    (1)
DSX$DOSUMMARY                   00000000-XR                   290     (1)
DSX$ENDPASS                     00000000-XR                   268     (1)
DSX$ENDSUB                      00000000-XR                   303     (1)
DSX$ERRDEV                      00000000-XR                   406     (1)
DSX$ERRHARD                     00000000-XR                   411     (1)
DSX$ERRPREP                     00000000-XR                   446     (1)
DSX$ERRSOFT                     0000000C-XR                   416     (1)
DSX$ERRSYS                      00000000-XR                   401     (1)
DSX$ESCAPE                      00000000-XR                   318     (1)
DSX$FREEDBGSYM                  00000000-XR                   563     (1)
DSX$GETADDRESS                  00000000-XR                   366     (1)
```

DSX$GETBUF	00000000-XR			465	(1)		
DSX$GETDATA	00000000-XR			354	(1)		
DSX$GETLOGICAL	00000000-XR			372	(1)		
DSX$GETSTRING	00000000-XR			378	(1)		
DSX$GETTERM	00000000-XR			573	(1)		
DSX$GETVIELD	00000000-XR			360	(1)		
DSX$GPHARD	00000000-XR			273	(1)		
DSX$HELP	00000000-XR			558	(1)		
DSX$INITSCB	00000000-XR			518	(1)		
DSX$INLOOP	00000000-XR			313	(1)		
DSX$LOAD	00000000-XR			544	(1)		
DSX$LOADPCS	00000000-XR			568	(1)		
DSX$MAPDBGBLOCK	00000000-XR			456	(1)		
DSX$MMOFF	00000000-XR			500	(1)		
DSX$MMON	00000000-XR			495	(1)		
DSX$MOUNT	00000000-XR			1809	(1)		
DSX$PARSE	00000000-XR			396	(1)		
DSX$PRINTB	00000000-XR			421	(1)		
DSX$PRINTF	00000000-XR			431	(1)		
DSX$PRINTS	00000000-XR			436	(1)		
DSX$PRINTSIG	00000000-XR			441	(1)		
DSX$PRINTX	00000000-XR			426	(1)		
DSX$PROBE	00000000-XR			549	(1)		
DSX$RELBUF	00000000-XR			470	(1)		
DSX$SERVICE_DISPATCHER	00000000-R	1853	(1)	578	(1)		
DSX$SETIPL	00000000-XR			523	(1)		
DSX$SETMAP	00000000-XR			533	(1)		
DSX$SETPRIEXV	00000000-XR			451	(1)		
DSX$SETVEC	00000000-XR			508	(1)		
DSX$SHOWCHAN	00000000-XR			539	(1)		
DSX$WAITMS	00000000-XR			329	(1)		
DSX$WAITUS	00000000-XR			334	(1)		
DS_ERRSUP	00000000-XR			1902	(1)		
ERL$DEVICERR	00000000-XR			1615	(1)		
ERL$DEVICTMO	00000000-XR			1617	(1)		
ERL$RELEASEMB	00000000-XR			1723	(1)		
EXE$ABORTIO	00000000-XR			1619	(1)		
EXE$ALLOC	00000000-XR			647	(1)		
EXE$ALLOCBUF	00000000-XR			1727	(1)		
EXE$ALLOCIRP	00000000-XR			1683	(1)		
EXE$ALONONPAGED	00000000-XR			1685	(1)		
EXE$ALTQUEPKT	00000000-XR			1773	(1)		
EXE$ASCTIM	00000000-XR			657	(1)		
EXE$ASSIGN	00000000-XR			662	(1)		
EXE$BINTIM	00000000-XR			667	(1)		
EXE$BUFFRQUOTA	00000000-XR			1731	(1)		
EXE$BUFQUOPRC	00000000-XR			1729	(1)		
EXE$CANCEL	00000000-XR			672	(1)	747	(1)
EXE$CANTIM	00000000-XR			677	(1)		
EXE$CARRIAGE	00000000-XR			1749	(1)		
EXE$CLREF	00000000-XR			707	(1)		
EXE$CNTREG	00000000-XR			712	(1)		
EXE$DASSGN	00000000-XR			752	(1)		
EXE$DEANONPAGED	00000000-XR			1687	(1)		
EXE$EXPREG	00000000-XR			817	(1)		
EXE$FAO	00000000-XR			822	(1)		
EXE$FAOL	00000000-XR			827	(1)		

```
EXE$FINISHIO                    00000000-XR                     1733    (1)
EXE$FINISHIOC                   00000000-XR                     1689    (1)
EXE$FORK                        00000000-XR                     1691    (1)
EXE$GETTIM                      00000000-XR                     847     (1)
EXE$GL_ABSTIM                   00000810-R    1598    (1)
EXE$GQ_SYSTIME                  00000B00-R    1787    (1)
EXE$GTCHAN                      00000000-XR                     1063    (1)     852     (1)
EXE$HIBER                       00000000-XR                     857     (1)
EXE$INSERTIRP                   00000000-XR                     1711    (1)
EXE$INSIOQ                      00000000-XR                     1783    (1)
EXE$IOFORK                      00000000-XR                     1621    (1)
EXE$MAXACMODE                   00000000-XR                     1794    (1)
EXE$MODIFY                      00000000-XR                     1745    (1)
EXE$MODIFYLOCK                  00000000-XR                     1747    (1)
EXE$MODIFYLOCKR                 00000C00-XR                     1693    (1)
EXE$NUMTIM                      00000000-XR                     888     (1)
EXE$ONEPARM                     00000000-XR                     1623    (1)
EXE$PWRTIMCHK                   00000000-XR                     1625    (1)
EXE$QIO                         00000000-XR                     613     (1)
EXE$QIO2                        =00000204-R   614     (1)       898     (1)
EXE$QIODRVPKT                   00000000-XR                     1695    (1)
EXE$QIORETURN                   00000000-XR                     1713    (1)
EXE$READ                        00000000-XR                     1735    (1)
EXE$READCHK                     00000000-XR                     1737    (1)
EXE$READCHKR                    00000000-XR                     1739    (1)
EXE$READEF                      00000000-XR                     904     (1)
EXE$READLOCK                    00000000-XR                     1796    (1)
EXE$READLOCKR                   00000000-XR                     1705    (1)
EXE$SENSEMODE                   00000000-XR                     1627    (1)
EXE$SETAST                      00000000-XR                     929     (1)
EXE$SETCHAR                     00000000-XR                     1629    (1)
EXE$SETEF                       00000000-XR                     934     (1)
EXE$SETIMR                      00000000-XR                     954     (1)
EXE$SETMODE                     00000000-XR                     1631    (1)
EXE$SETPRT                      00000000-XR                     965     (1)
EXE$SNDEVMSG                    00000000-XR                     1633    (1)
EXE$UNWIND                      00000000-XR                     1008    (1)
EXE$WAITFR                      00000000-XR                     1013    (1)
EXE$WAKE                        00000000-XR                     1018    (1)
EXE$WFLAND                      00000000-XR                     1023    (1)
EXE$WFLOR                       00000000-XR                     1028    (1)
EXF$WRITE                       00000000-XR                     1741    (1)
EXE$WRITECHK                    00000000-XR                     1703    (1)
EXE$WRITECHKR                   00000000-XR                     1743    (1)
EXE$WRITELOCK                   00000000-XR                     1697    (1)
EXE$WRITELOCKR                  00000000-XR                     1707    (1)
EXE$ZEROPARM                    00000000-XR                     1635    (1)
FAKE_JUMP                       00000032-R    1904    (1)       323     (1)
IOC$ALLOSPT                     00000000-XR                     1792    (1)
IOC$ALOUBAMAP                   00000000-XR                     1719    (1)
IOC$ALOUBAMAPN                  00000000-XR                     1725    (1)
IOC$ALTUBAMAP                   00000000-XR                     1721    (1)
IOC$APPLYECC                    00000000-XR                     1637    (1)
IOC$CANCELIO                    00000000-XR                     1639    (1)
IOC$DIAGBUFILL                  00000000-XR                     1641    (1)
IOC$GETBYTE                     00000000-XR                     1755    (1)
IOC$GL_PSBL                     0000081C-R    1601    (1)
```

```
IOC$GL_PSFL                     00000818-R   1600   (1)      1600   (1)      1601   (1)
IOC$INITBUFWIND                 00000000-XR                  1759   (1)
IOC$INITIATE                    00000000-XR                  1709   (1)
IOC$LOADMBAMAP                  00000000-XR                  1643   (1)
IOC$LOADUBAMAP                  00000000-XR                  1645   (1)
IOC$LOADUBAMAPA                 00000000-XR                  1753   (1)
IOC$MOVFRUSER                   00000000-XR                  1790   (1)
IOC$MOVFRUSER1                  00000000-XR                  1763   (1)
IOC$MOVFRUSER2                  00000000-XR                  1761   (1)
IOC$MOVTOUSER                   00000000-XR                  1647   (1)
IOC$MOVTOUSER1                  00000000-XR                  1767   (1)
IOC$MOVTOUSER2                  00000000-XR                  1765   (1)
IOC$PTETOPFN                    00000000-XR                  1785   (1)
IOC$PURGDATAP                   00000000-XR                  1649   (1)
IOC$PUTBYTE                     00000000-XR                  1757   (1)
IOC$RELCHAN                     00000000-XR                  1651   (1)
IOC$RELDATAP                    00000000-XR                  1653   (1)
IOC$RELMAPREG                   00000000-XR                  1655   (1)
IOC$RELSCHAN                    00000000-XR                  1657   (1)
IOC$REQCOM                      00000000-XR                  1659   (1)
IOC$REQDATAP                    00000000-XR                  1661   (1)
IOC$REQDATAPNW                  00000000-XR                  1715   (1)
IOC$REQMAPREG                   00000000-XR                  1663   (1)
IOC$REQPCHANH                   00000000-XR                  1665   (1)
IOC$REQPCHANL                   00000000-XR                  1667   (1)
IOC$REQSCHANH                   00000000-XR                  1751   (1)
IOC$REQSCHANL                   00000000-XR                  1669   (1)
IOC$RETURN                      00000000-XR                  1671   (1)
IOC$RTN                         00000930-R   1671   (1)
IOC$SENSEDISK                   00000000-XR                  1673   (1)
IOC$UPDATRANSP                  00000000-XR                  1675   (1)
IOC$WFIKPCH                     00000000-XR                  1677   (1)
IOC$WFIRLCH                     00000000-XR                  1679   (1)
KB_CHECK                        00000000-XR                  1905   (1)
LEAP_APT                        000001F4-R   601    (1)    #-255    (1)
LEAP_BOOT                       000001EC-R   595    (1)    #-251    (1)
LF                              =0000000A    192    (1)     204     (1)
MMG$GL_POBASE                   00000808-R   1594   (1)
MMG$GL_SPTBASE                  00000804-R   1592   (1)
MT$CHECK_ACCESS                 00000000-XR                  1681   (1)
POPT_BASE                       00000000-XR                  1595   (1)
RESRVD                          000001FA-R   604    (1)    #-594    (1)     #-599    (1)
RESRVD_ENTRY                    0000001D-R   1901   (1)     1033    (1)      1038    (1)      1043    (1)      1048    (1)
                                                            1053    (1)      1058    (1)      1068    (1)      1073    (1)
                                                            1078    (1)      1083    (1)      1088    (1)      1093    (1)
                                                            1098    (1)      1103    (1)      1108    (1)      1113    (1)
                                                            1118    (1)      1123    (1)      1128    (1)      1133    (1)
                                                            1138    (1)      1143    (1)      1148    (1)      1153    (1)
                                                            1158    (1)      1163    (1)      1168    (1)      1173    (1)
                                                            1183    (1)      1193    (1)      1198    (1)      1203    (1)
                                                            1208    (1)      1223    (1)      1233    (1)      1238    (1)
                                                            1243    (1)      1248    (1)      1253    (1)      1258    (1)
                                                            1269    (1)      1274    (1)      1279    (1)      1284    (1)
                                                            1289    (1)      1294    (1)      1299    (1)      1304    (1)
                                                            1309    (1)      1314    (1)      1319    (1)      1324    (1)
                                                            1329    (1)      1334    (1)      1339    (1)      1344    (1)
                                                            1349    (1)      1354    (1)      1359    (1)      1364    (1)
```

				1369	(1)	1374	(1)	1379	(1)	1384	(1)
				1389	(1)	1394	(1)	1399	(1)	1404	(1)
				1409	(1)	1414	(1)	1419	(1)	1424	(1)
				1429	(1)	1434	(1)	1439	(1)	1444	(1)
				1449	(1)	1454	(1)	1459	(1)	1464	(1)
				1469	(1)	1474	(1)	1479	(1)	1484	(1)
				1489	(1)	1494	(1)	1499	(1)	1504	(1)
				1509	(1)	1514	(1)	1519	(1)	1524	(1)
				1529	(1)	1534	(1)	1539	(1)	1544	(1)
				1549	(1)	1554	(1)	1559	(1)	1564	(1)
				1569	(1)	1574	(1)	260	(1)	475	(1)
				480	(1)	485	(1)	490	(1)	584	(1)
				589	(1)	604	(1)	627	(1)	632	(1)
				637	(1)	642	(1)	652	(1)	682	(1)
				687	(1)	692	(1)	697	(1)	702	(1)
				717	(1)	722	(1)	727	(1)	732	(1)
				737	(1)	742	(1)	757	(1)	762	(1)
				767	(1)	772	(1)	777	(1)	782	(1)
				787	(1)	792	(1)	797	(1)	802	(1)
				807	(1)	832	(1)	837	(1)	842	(1)
				862	(1)	867	(1)	878	(1)	883	(1)
				893	(1)	909	(1)	914	(1)	919	(1)
				924	(1)	939	(1)	944	(1)	949	(1)
				976	(1)	981	(1)	986	(1)	991	(1)
RMS$CLOSE	00000000-XR			1213	(1)						
RMS$CONNECT	00000000-XR			1218	(1)						
RMS$DISCONNECT	00000000-XR			1228	(1)						
RMS$GET	00000000-XR			1178	(1)						
RMS$OPEN	00000000-XR			1263	(1)						
RMS$READ	00000000-XR			1188	(1)						
SCH$GL_PCBVEC	0000080C-R	1596	(1)								
SCH$LOCKW	00000000-XR			1769	(1)						
SCH$POSTEF	00000000-XR			1699	(1)						
SCH$QAST	00000000-XR			1701	(1)						
SCH$UNLOCK	00000000-XR			1771	(1)						
SYS$ALLOC	00000238-R	645	(1)								
SYS$ASCTIM	00000248-R	655	(1)								
SYS$ASSIGN	00000250-R	660	(1)								
SYS$BINTIM	00000258-R	665	(1)								
SYS$CALL_HANDL	00000210-R	620	(1)								
SYS$CANCEL	00000260-R	670	(1)								
SYS$CANTIM	00000268-R	675	(1)								
SYS$CLOSE	000005B8-R	1211	(1)								
SYS$CLREF	00000298-R	705	(1)								
SYS$CNTREG	000002A0-R	710	(1)								
SYS$CONNECT	000005C0-R	1216	(1)								
SYS$DALLOC	000002D8-R	745	(1)								
SYS$DASSGN	000002E0-R	750	(1)								
SYS$DISCONNECT	000005D0-R	1226	(1)								
SYS$EXIT	00000340-R	810	(1)								
SYS$EXPREG	00000348-R	815	(1)								
SYS$FAO	00000350-R	820	(1)								
SYS$FAOL	00000358-R	825	(1)								
SYS$GET	00000580-R	1176	(1)								
SYS$GETCHN	000004C8-R	1061	(1)								
SYS$GETTIM	00000378-R	845	(1)								
SYS$GL_OPRMBX	00000800-R	1590	(1)								

```
SYS$GTCHAN                      00000380-R    850     (1)
SYS$HIBER                       00000388-R    855     (1)
SYS$LKWSET                      000003A0-R    870     (1)
SYS$NUMTIM                      000003B8-R    886     (1)
SYS$OPEN                        00000608-R    1261    (1)
SYS$QIO                         000003C8-R    896     (1)
SYS$QIOW                        00000200-R    611     (1)
SYS$READ                        00000590-R    1186    (1)
SYS$READEF                      000003D0-R    902     (1)
SYS$SETAST                      000003F8-R    927     (1)
SYS$SETEF                       00000400-R    932     (1)
SYS$SETIMR                      00000420-R    952     (1)
SYS$SETPRI                      00000428-R    957     (1)
SYS$SETPRT                      00000430-R    963     (1)
SYS$SETRWM                      00000438-R    968     (1)
SYS$TRNLOG                      00000458-R    989     (1)
SYS$ULKPAG                      00000460-R    994     (1)
SYS$ULWSET                      00000468-R    1000    (1)
SYS$UNWIND                      00000470-R    1006    (1)
SYS$WAITFR                      00000478-R    1011    (1)     #-616     (1)
SYS$WAKE                        00000480-R    1016    (1)
SYS$WFLAND                      00000488-R    1021    (1)
SYS$WFLOR                       00000490-R    1026    (1)
T_RESRVD                        00000006-R    203     (1)       1902    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$PUSHADR	1	1902 (1)	1902 (1)
ERRSUP_S	1	1902 (1)	1902 (1)
MODNAM	1	201 (1)	201 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	26	00:00:00.11	00:00:00.20
Command processing	110	00:00:00.77	00:00:01.56
Pass 1	395	00:00:05.77	00:00:08.13
Symbol table sort	1	00:00:00.40	00:00:00.44
Pass 2	361	00:00:03.46	00:00:05.37
Symbol table output	39	00:00:00.27	00:00:00.63
Psect synopsis output	4	00:00:00.04	00:00:00.12
Cross-reference output	80	00:00:01.23	00:00:02.19
Assembler run totals	1020	00:00:12.06	00:00:18.65

The working set limit was 1000 pages.
25604 bytes (51 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 321 non-local and 1 local symbols.
1908 source lines were read in Pass 1, producing 0 object records in Pass 2.
3 pages of virtual memory were used to define 3 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	0
DRB1:[DS.WORK]DS.MLB;218	2
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	1
TOTALS (all libraries)	3

18 GETS were required to define 3 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ENTRY/UPDA=(ENTRY.UPD,ENTRY.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

Table of contents

(1) 138 Libraries, Macros, Equated Symbols
(1) 165 Work Psect Declarations
(1) 179 Data Psect Declarations
(1) 393 DSX$ErrSoft Routine
(1) 445 DSX$ErrHard Routine
(1) 497 DSX$ErrPrep Routine [17]
(1) 549 DSX$ErrDev Routine
(1) 604 DSX$ErrSys Routine
(1) 659 RRtpError Routine
(1) 807 DS_TestID Routine
(1) 833 DS_ErrSup Routine
(1) 957 DSR$Completion Routine

```
0000      1          .Title  ERROR   *** ERROR Error routines
0000      2          .Ident  /07-24/
0000      3          .NoShow Conditionals
0000      4
0000      5 ;++
0000      6 ; Copyright (c) 1977, 1983, 1984
0000      7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      8 ;
0000      9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     15 ; REMAIN IN DEC.
0000     16 ;
0000     17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     19 ; CORPORATION.
0000     20 ;
0000     21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     23 ;--
```

```
0000     25 ;++
0000     26 ; FACILITY:
0000     27 ;
0000     28 ;       VAX Diagnostic Supervisor
0000     29 ;
0000     30 ; ABSTRACT:
0000     31 ;
0000     32 ; ENVIRONMENT:
0000     33 ;
0000     34 ; AUTHOR:
0000     35 ;
0000     36 ;       Tom Soutter     16-NOV-77       VERSION 01
0000     37 ;
0000     38 ; MODIFIED BY:
0000     39 ;                       TOM SOUTTER     08-DEC-77       VERSION 02 (ESSAA-3.05)
0000     40 ;       01              DSPR #20.  "ERROR HALT AT ..." MESSAGE INSERTED.
0000     41 ;
0000     42 ;                       KEN CHAPMAN     30-DEC-77       VERSION 03 (ESSAA-3.06)
0000     43 ;       02              PASS AP TO USER'S ERROR ROUTINE.
0000     44 ;                       NICK HOWGATE    06-DEC-78       VERSION 04 (ESSAA-5.01)
0000     45 ;       03              EMULATE CHARACTER STRING INSTRUCTIONS
0000     46 ;       04              Removed DSA$V_ERR_LOOP, use DS$INLOOP instead
0000     47 ;
0000     48 ;                       Dave Butenhof, 10-sep-1980, Version 6.0
0000     49 ;       05              Add error text to support RMS error codes used within
0000     50 ;                       supervisor.
0000     51 ;
0000     52 ;                       Dave Butenhof, 30-sep-1980, Version 6.1
0000     53 ;       06              Change forced word-relative references to longword-relative,
0000     54 ;                       since supervisor has g own.
0000     55 ;
0000     56 ;       07              - Jack Stansbury, 21-Oct-1981, Version 6.-
0000     57 ;                       Added calls to QA$MAIN for the QA enhancement.
0000     58 ;                       Also added .LIBRARY statements for $DS and $DIAG.
0000     59 ;                       Also changed some of the error messages to lower case.
0000     60 ;
0000     61 ;       08              - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     62 ;                       Fixed truncation errors.
0000     63 ;
0000     64 ;       09              - Jack Stansbury, 12-Nov-1981, Version 6.-
0000     65 ;                       Took out references to HALTD and HALTI. Changed them to HALT.
0000     66 ;
0000     67 ;       10              - Dave Butenhof, 17-Nov-1981, Version 6.5
0000     68 ;                       Add 'end of error' message, and typecodes
0000     69 ;
0000     70 ;       11              - Jack Stansbury, 21-Nov-1981, Version 6.5
0000     71 ;                       Commented out several of the calls to QA_MAIN because it
0000     72 ;                       appears as though they will not be needed for the 6.5 DS version.
0000     73 ;
0000     74 ;       12              - Dave Butenhof, 25-Nov-1981, version 6.5
0000     75 ;                       Added error text for error '54', 'Fatal controller error',
0000     76 ;                       as it is appallingly common.
0000     77 ;
0000     78 ;       13              - Dave Butenhof, 15-Dec-1981, Version 6.6
0000     79 ;                       Fix error trailer message to only type out if IE1 isn't
0000     80 ;                       set.
0000     81 ;
```

```
                                0000     82 ;        14      - Jack Stansbury, 17-Dec-1981, Version ?.?
                                0000     83 ;                Added all the registers to the register masks on each of the
                                0000     84 ;                error routines. These are for the QA routines.
                                0000     85 ;
                                0000     86 ;        15      - Dave Butenhof, 29-Dec-1981, Version 6.6
                                0000     87 ;                Use new error counters for each class of error (hard,
                                0000     88 ;                soft, etc.).
                                0000     89 ;
                                0000     90 ;        16      - Jack Stansbury, 25-Jan-1982, Version 6.6
                                0000     91 ;                Discovered that the error number is truncated to a word in the
                                0000     92 ;                RRprError error reporting routine even though the error number
                                0000     93 ;                that is passed to the routine is a longword in length. This is
                                0000     94 ;                most likely because of APT running on PDP-11's, which operate
                                0000     95 ;                on word lengths. So, I changed one instruction to zero out the
                                0000     96 ;                top word in DSA$GL_ERRNO when the error number is moved to this
                                0000     97 ;                longword.
                                0000     98 ;
                                0000     99 ;        17      Marion Baggett, 2-apr-1982, Version 6.7
                                0000    100 ;                added the PREPERR error routine for reporting errors caused by a
                                0000    101 ;                testing starting and the hardware not being properly set up.
                                0000    102 ;
                                0000    103 ;        18      Marion Baggett, 9-Apr-1982, Version 6.7
                                0000    104 ;                Change DS$PRINTF to DSX$PRINT for type coding
                                0000    105 ;
                                0000    106 ;        19      Jack Stansbury, 10-Apr-1982, Version 6.7
                                0000    107 ;                Changed some of the BBC instructions to the new DS macros such
                                0000    108 ;                as Br_If_ ....
                                0000    109 ;
                                0000    110 ;        20      Jack Stansbury, 11-May-1982, Version 6.8
                                0000    111 ;                Added code to store the MsgAdr parameter from the $DS_ERRxxx
                                0000    112 ;                macros into a location for use by the CRD routines.
                                0000    113 ;
                                0000    114 ;        21      Jack Stansbury, 21-Dec-1982, Version 6.11
                                0000    115 ;                Printed out some more flags, locations, etc. in the ErrSup
                                0000    116 ;                routine.
                                0000    117 ;
                                0000    118 ;        22      Jack Stansbury, 03-Mar-1983, Version 6.11
                                0000    119 ;                Changed the code that sets the DS$GA_User_Error_Text so that
                                0000    120 ;                this location is set ONLY when CRD is running. This MUST be
                                0000    121 ;                true or else the CRD dispatch vectors will get scrwed up.
                                0000    122 ;                The condition was this: if a diagnostic was run that issued
                                0000    123 ;                an error (prep, sys, dev, hard, or soft), then if CRD was run,
                                0000    124 ;                and one of the diags run by CRD encountered a prep error, the
                                0000    125 ;                path from the MEN$Preparation_Error_Text routine to the address
                                0000    126 ;                of the user's error text would be lost, and an exception would
                                0000    127 ;                result.
                                0000    128 ;
                                0000    129 ;        23      John Ciukaj     4-Apr-1983      Version 6.11
                                0000    130 ;                - Changed all references to CRD Bits in DSA Flags
                                0000    131 ;                  distinguishing between Online and Offline.
                                0000    132 ;
                                0000    133 ;        24      Bob Bergazzi    24-Apr-1984     Version 7.0
                                0000    134 ;                Fixed edit # 21, FMTERRSUP1, which pushed contents instead of the
                                0000    135 ;                address of DS$GT_VDS_VERSION on the stack.
                                0000    136 ;--
```

```
                        0000   138          .SBTTL  Libraries, Macros, Equated Symbols
                        0000   139 ;
                        0000   140 ; INCLUDE FILES:
                        0000   141 ;
                        0000   142          .LIBRARY        /SYS$LIBRARY:LIB/       ;                               [17]
                        0000   143          .LIBRARY        /$DS/                   ;                               [07]
                        0000   144          .LIBRARY        /$DIAG/                 ;                               [07]
                        0000   145
                        0000   146 ;
                        0000   147 ; EQUATED SYMBOLS:
                        0000   148 ;
                        0000   149
00000003                0000   150          BPTCODE     = 03                    ; BREAKPOINT OPCODE
00000020                0000   151          TESTID_LEN = 32                     ; TEXT BUFFER FOR TEST/SUBTEST MESSAGE
                        0000   152
                        0000   153          APTDEF                              ; APT MESSAGE CODE DEFINITIONS
                        0000   154          DSFDEF                              ; CONTROL FLAG DEFINITIONS
                        0000   155          DSQA                                ; QA routine name constants             [07]
                        0000   156          $DS_CFDEF
                        0000   157          $DS_DSDEF
                        0000   158          $DS_DSADEF                          ; CONTROL FLAG DEFINITIONS
                        0000   159          $DS_ERRDEF
                        0000   160          $DS_HDRDEF                          ; DIAGNOSTIC PROGRAM HEADER SYMBOLS
                        0000   161          $DS_HPODEF                          ; P-TABLE OFFSET DEFINITIONS
                        0000   162          $RMSDEF                             ; RMS return code definitions
                        0000   163          $ds_typedef                         ; Define error typecodes                [10]
```

```
                                0000   165          .Subtitle       Work Psect Declarations
                                0000   166 ;
                                0000   167 ; OWN STORAGE:
                                0000   168 ;
                                0000   169
                             00000000  170          .Psect  Work, Noshr, Noexe, Wrt, Long
                                0000   171
                                0000   172 Q_BUFQWD:                                  ; QUADWORD DESCRIPTOR
                    00000008    0000   173          .BLKQ   1                          ;  FOR MISC. STRINGS
                                0008   174 DS$GQ_TESTID::
                    00000010    0008   175          .BLKQ   1                          ; STRING LENGTH & POINTER
                                0010   176 T_TESTID:
                    00000030    0010   177          .BLKB   TESTID_LEN                 ; BUFFER FOR TEST/SUBTEST MESSAGE
```

```
                                        0030   179          .Subtitle       Data Psect Declarations
                                    00000000   180          .Psect  Data, Shr, Noexe, Ncwrt, Byte
                                        0000   181
                                        0000   182 MSGBELL:
                                  07    0000   183          .ASCII  <7>                         ; ASCII 'BELL'
                                        0001   184 HARDER:
                                  06    0001   185          .byte   ds$k_type_errhard           ; Type code byte                    [10]
   72 6F 72 72 65 20 64 72 61 48 00'   0002   186          .ASCIC  .Hard error.
                                  0A    0002
                                        000D   187 SOFTER:
                                  07    000D   188          .byte   ds$k_type_errsoft           ; Type code byte                    [10]
   72 6F 72 72 65 20 74 66 6F 53 00'   000E   189          .ASCIC  .Soft error.
                                  0A    000E
                                        0019   190 DEVFER:
                                  08    C019   191          .byte   ds$k_type_errdev            ; Type code byte                    [10]
61 74 61 66 20 65 63 69 76 65 44 00'  001A   192          .ASCIC  .Device fatal error.        ;                                   [10]
                  72 6F 72 72 65 20 6C  0026
                                  12    001A
                                        002D   193 PREPER:
                                  1B    002D   194          .byte   ds$K_type_errprep           ; Type code byte                    [17]
20 74 6F 6E 20 65 63 69 76 65 44 00'  002E   195          .ASCIC  .Device not prepared error.     ;                               [17]
72 72 65 20 64 65 72 61 70 65 72 70   003A
                              72 6F   0046
                                  19    002E
                                        0048   196 SYSFER:
                                  05    0048   197          .byte   ds$k_type_errsys            ; Type code byte                    [10]
61 74 61 66 20 6D 65 74 73 79 53 00'  0049   198          .ASCIC  .System fatal error.        ;                                   [10]
                  72 6F 72 72 65 20 6C  0055
                                  12    0049
                                        005C   199 ersuper:                                    ; Supervisor error text             [10]
65 64 20 65 72 61 77 74 66 6F 53 00'  005C   200          .ascic  .Software detected error.       ;                               [10]
72 6F 72 72 65 20 64 65 74 63 65 74   0068
                                  17    005C
                                        0074   201 ANULL:
                                  00'   0074   202          .ASCIC  ..
                                  00    0074
                                        0075   203 QUNKDEV:
77 6F 6E 6B 6E 55 0000007D'010E0000'  0075   204          .ASCID  .Unknown device.
                  65 63 69 76 65 64 20 6E 0083
                                        008B   205 QINITSEC:
61 69 74 69 6E 49 00000093'010E0000'  008B   206          .ASCID  .Initialization section.
63 65 73 20 6E 6F 69 74 61 7A 69 6C   0099
                              6E 6F 69 74 00A5
                                        00A9   207 QCLEANSEC:
75 6E 61 65 6C 43 000000B1'010E0000'  00A9   208          .ASCID  .Cleanup section.
                  6E 6F 69 74 63 65 73 20 70  00B7
                                        00C0   209 QFMTTESTID:
21 20 74 73 65 74 000000C8'010E0000'  00C0   210          .ASCID  .test !UL, subtest !UL.
20 74 73 65 74 62 75 73 20 2C 4C 55   00CE
                                 4C 55 21   00DA
```

```
                                                00DD   212 FMTERRHLT:
6F 20 74 6C 61 48 20 2E 2E 2F 21 00'            00DD   213          .ASCIC  "!/.. Halt on error at PC !XL(X)!/"     ;                [07]
50 20 74 61 20 72 6F 72 72 65 20 6E             00E9
      2F 21 29 58 28 4C 58 21 20 43             00F5
                                  21            00DD
                                                00FF   214 FMTERRHDR:
20 2A 2A 2A 2A 2A 2A 2A 2A 2F 21 00'            00FF   215          .ASCIC  \!/********  !AC - !UL.!UL  ********\ -
21 2E 4C 55 21 20 2D 20 43 41 21 20             010B
2A 2A 2A 2A 2A 2A 2A 2A 20 20 4C 55             0117
2C 4C 55 21 20 73 73 61 50 20 2F 21             0123
20 72 6F 72 72 65 20 2C 53 41 21 20             012F
            44 25 21 20 2C 4C 55 21             013B
                                                0143   216                  "!/ Pass !UL, !AS, error !UL, !%D"-     ;                [07]
65 6C 69 68 77 20 43 41 21 20 2F 21             0143   217                  "!/ !AC while testing !AS: !AC!/"       ;                [07]
53 41 21 20 67 6E 69 74 73 65 74 20             014F
            2F 21 43 41 21 20 3A                015B
                                  62            00FF
                                                0162   218 fmtenderror:
20 2A 2A 2A 2A 2A 2A 2A 2A 2F 21 00'            0162   219          .ascic  \!/********  End of !AC number !UL ********!/!/\        ;        [10]
20 43 41 21 20 66 6F 20 64 6E 45 20             016E
2A 20 4C 55 21 20 72 65 62 6D 75 6E             017A
   2F 21 2F 21 2A 2A 2A 2A 2A 2A 2A             0186
                                  2E            0162
                                                0191   220 FMTERRSUP:
61 77 74 66 6F 53 20 3F 3F 2F 21 00'            0191   221          .ASCIC  '!/?? Software detected error in "!AC", error #!UL, !%D'-  ;    [07]
20 64 65 74 63 65 74 65 64 20 65 72             019D
41 21 22 20 6E 69 20 72 6F 72 72 65             01A9
21 23 20 72 6F 72 72 65 20 2C 22 43             01B5
            44 25 21 20 2C 4C 55                01C1
         2F 21 43 41 21 20 2F 21                01C8   222                  "!/ !AC!/"
                                  3E            0191
                                                01D0   223 FmtErrSup1:                                                          ;        [21]
67 61 6C 46 5F 4C 47 24 41 53 44 00'            01D0   224          .Ascic  'DSA$GL_Flags: !XL(X)  DS$GL_Flags: !XL(X)  Version: !AC!/' ;    [21]
44 20 20 29 58 28 4C 58 21 20 3A 73             01DC
20 3A 73 67 61 6C 46 5F 4C 47 24 53             01E8
73 72 65 56 20 20 29 58 28 4C 58 21             01F4
      2F 21 43 41 21 20 3A 6E 6F 69             0200
                                  39            01D0
                                                020A   225
                                                020A   226 FMTERRSUP2:
20 20 4C 58 21 20 3A 43 50 5F 21 00'            020A   227          .ASCIC  "!_PC: !XL  PSL: !XL!/"-
      2F 21 4C 58 21 20 3A 4C 53 50             0216
52 20 20 4C 58 21 20 3A 30 52 5F 21             0220   228                  "!_R0: !XL  R1:  !XL  R2:  !XL  R3:  !XL!/"-
3A 32 52 20 20 4C 58 21 20 20 3A 31             022C
20 20 3A 33 52 20 20 4C 58 21 20 20             0238
                  2F 21 4C 58 21                0244
52 20 20 4C 58 21 20 3A 34 52 5F 21             0249   229                  "!_R4: !XL  R5:  !XL  R6:  !XL  R7:  !XL!/"-
3A 36 52 20 20 4C 58 21 20 20 3A 35             0255
20 20 3A 37 52 20 20 4C 58 21 20 20             0261
                  2F 21 4C 58 21                026D
52 20 20 4C 58 21 20 3A 38 52 5F 21             0272   230                  "!_R8: !XL  R9:  !XL  R10: !XL  R11: !XL!/"
30 31 52 20 20 4C 58 21 20 20 3A 39             027E
20 3A 31 31 52 20 20 4C 58 21 20 3A             028A
                  2F 21 4C 58 21                0296
                                  90            020A
                                                0298   231 FMTERRSUP3:
39 28 37 21 4C 58 21 3A 4C 58 21 00'            0298   232          .ASCIC  "!XL:!XL!7(9XL)!/"
```

```
                    2F 21 29 4C 58  02A7
                                10  029B
```

```
                                02AC    234 T_ERROR_MASK:
21 2D 44 41 21 2D 43 41 21 2F 21 00' 02AC    235         .ASCIC  '!/!AC-!AD-!AC (R0=!XL)!/'
21 29 4C 58 21 3D 30 52 28 20 43 41  02B8
                                2F   02C4
                                18   02AC
                                02C5    236 UTILITY_CODES:
                        00000004'       02C5    237         .LONG   <10$-.>/4
                        00000000        02C9    238         .LONG   0
                        00000001        02CD    239         .LONG   1
                        00000066        02D1    240         .LONG   ^X66
                                02D5    241 10$:
                                02D5    242 UTILITY_TEXT:
                            0000        02D5    243         .WORD   0
                            0006'       02D7    244         .WORD   10$-.
                            000C'       02D9    245         .WORD   20$-.
                            000F'       02DB    246         .WORD   30$-.
                  4D 45 54 53 59 53 25 00' 02DD 247 10$:    .ASCIC  '%SYSTEM'
                                07   02DD
                     53 4D 52 25 00' 02E5    248 20$:    .ASCIC  '%RMS'
                                04   02E5
                  47 41 49 44 25 00' 02EA    249 30$:    .ASCIC  '%DIAG'
                                05   02EA
                                02F0    250
                                02F0    251 ERROR_CODES:
                        0000002D'       02F0    252         .LONG   <10$-.>/4
                        00000000'       02F4    253         .LONG   SS$_ACCVIO              ; Access violation
                        00000000'       02F8    254         .LONG   SS$_NOSUCHDEV           ; No such device
                        00000000'       02FC    255         .LONG   SS$_NOSUCHFILE          ; No such file
                        00000000'       0300    256         .long   ss$_ctrlerr             ; Fatal controller error        [12]
                        00018290        0304    257         .LONG   RMS$_FNF&^XFFFFFFF8     ; RMS no such file
                        000184C0        0308    258         .LONG   RMS$_DEV&^XFFFFFFF8     ; RMS bad device
                        0001C048        030C    259         .LONG   RMS$_DNF&^XFFFFFFF8     ; RMS directory not found
                        0001C000        0310    260         .LONG   RMS$_ACC&^XFFFFFFF8     ; RMS file access error
                        00018490        0314    261         .LONG   RMS$_CCR&^XFFFFFFF8     ; RMS connect error
                        000184D0        0318    262         .LONG   RMS$_DME&^XFFFFFFF8     ; RMS pool allocation error
                        00018278        031C    263         .LONG   RMS$_EOF&^XFFFFFFF8     ; RMS end of file
                        00018508        0320    264         .LONG   RMS$_FAB&^XFFFFFFF8     ; RMS invalid FAB
                        00018560        0324    265         .LONG   RMS$_IFI&^XFFFFFFF8     ; RMS invalid internal file index
                        00018570        0328    266         .LONG   RMS$_IOP&^XFFFFFFF8     ; RMS illegal operation
                        00018580        032C    267         .LONG   RMS$_ISI&^XFFFFFFF8     ; RMS invalid RAB/FAB pair
                        00018638        0330    268         .LONG   RMS$_RAB&^XFFFFFFF8     ; RMS invalid RAB
                        0001C0F0        0334    269         .LONG   RMS$_RER&^XFFFFFFF8     ; RMS file read error
                        000181A8        0338    270         .LONG   RMS$_RTB&^XFFFFFFF8     ; RMS record truncation warning
                        00018658        033C    271         .LONG   RMS$_RFA&^XFFFFFFF8     ; RMS invalid RFA
                        00018578        0340    272         .LONG   RMS$_IRC&^XFFFFFFF8     ; RMS invalid record
                        00018528        0344    273         .LONG   RMS$_FNM&^XFFFFFFF8     ; FNM error in filename
                        00660008        0348    274         .LONG   DS$_OVERFLOW            ; Field overflow
                        00660010        034C    275         .LONG   DS$_NULLSTR             ; Null input string
                        00660018        0350    276         .LONG   DS$_ILLCHAR             ; Illegal character
                        00660020        0354    277         .LONG   DS$_PROGERR             ; Programmer error
                        00660028        0358    278         .LONG   DS$_TRUNCATE            ; Data truncated
                        00660038        035C    279         .LONG   DS$_IVVECT              ; Invalid vector
                        00660040        0360    280         .LONG   DS$_IVADDR              ; Invalid vector address
                        00660048        0364    281         .LONG   DS$_VASFULL             ; virtural address space full
                        00660050        0368    282         .LONG   DS$_INSFMEM             ; Insufficient memory
                        00660060        036C    283         .LONG   DS$_IHWE                ; Initial channel hardware error
                        00660068        0370    284         .LONG   DS$_FHWE                ; Final channel hardware error
```

```
                00660070  0374    285          .LONG   DS$_LOGIC               ; Program logic error
                00660078  0378    286          .LONG   DS$_ILLPAGCNT           ; Illegal page count
                00660080  037C    287          .LONG   DS$_FRAGBUF             ; Buffer is fragmented
                00660088  0380    288          .LONG   DS$_MCHK                ; Machine check
                00660090  0384    289          .LONG   DS$_KRNLSTK             ; Kernel stack not valid
                00660098  0388    290          .LONG   DS$_POWER               ; Power fail interrupt
                006600A0  038C    291          .LONG   DS$_TRANSL              ; Translation not valid
                006600B0  0390    292          .LONG   DS$_NOTIMP              ; Not implemented
                006600B8  0394    293          .LONG   DS$_IPL2HI              ; IPL too high
                006600C0  0398    294          .LONG   DS$_ICERR               ; Interval clock error
                006600D0  039C    295          .LONG   DS$_ARITH               ; Arithmetic trap
                006600D8  03A0    296          .LONG   DS$_UNEXPINT            ; Unexpected interrupt
                          03A4    297 10$:
                          03A4    298 ERROR_TEXT:
                    005A' C3A4    299          .WORD   0$-.                    ; Unknown error
                    00F4' 03A6    300          .WORD   60$-.                   ; ^X0C Access violation
                    0064' 03A8    301          .WORD   10$-.
                    007C' 03AA    302          .WORD   20$-.
                    04C8' 03AC    303          .word   440$-.                  ; Fatal controller error        [12]
                    0091' 03AE    304          .WORD   30$-.
                    00A3' 03B0    305          .WORD   40$-.
                    00CF' 03B2    306          .WORD   50$-.
                    02C4' 03B4    307          .WORD   300$-.
                    02DE' 03B6    308          .WORD   310$-.
                    02F4' 03B8    309          .WORD   320$-.
                    0310' 03BA    310          .WORD   330$-.
                    0328' 03BC    311          .WORD   340$-.
                    034D' 03BE    312          .WORD   350$-.
                    037D' 03C0    313          .WORD   360$-.
                    03B2' 03C2    314          .WORD   370$-.
                    03E4' 03C4    315          .WORD   380$-.
                    0409' 03C6    316          .WORD   390$-.
                    041C' 03C8    317          .WORD   400$-.
                    0442' 03CA    318          .WORD   410$-.
                    0469' 03CC    319          .WORD   420$-.
                    048F' 03CE    320          .WORD   430$-.
                    00E0' 03D0    321          .WORD   70$-.
                    00ED' 03D2    322          .WORD   80$-.
                    00FD' 03D4    323          .WORD   90$-.
                    010D' 03D6    324          .WORD   100$-.
                    011C' 03D8    325          .WORD   110$-.
                    0129' 03DA    326          .WORD   120$-.
                    0136' 03DC    327          .WORD   130$-.
                    014B' 03DE    328          .WORD   140$-.
                    0165' 03E0    329          .WORD   150$-.
                    0177' 03E2    330          .WORD   160$-.
                    0194' 03E4    331          .WORD   170$-.
                    01AF' 03E6    332          .WORD   180$-.
                    01C1' 03E8    333          .WORD   190$-.
                    01D2' 03EA    334          .WORD   200$-.
                    01E5' 03EC    335          .WORD   210$-.
                    01F1' 03EE    336          .WORD   220$-.
                    0206' 03F0    337          .WORD   230$-.
                    0219' 03F2    338          .WORD   240$-.
                    022D' 03F4    339          .WORD   250$-.
                    023B' 03F6    340          .WORD   260$-.
                    0246' 03F8    341          .WORD   270$-.
```

```
                                        0259' 03FA    342          .WORD   280$-.
                                        0267' 03FC    343          .WORD   290$-.
72 72 65 20 6E 77 6F 6E 6B 6E 55 00'    03FE    344 0$:      .ASCIC  "Unknown error"
                                  72 6F 040A
                                     0D 03FE
20 2C 56 45 44 48 43 55 53 4F 4E 00'    040C    345 10$:     .ASCIC  "NOSUCHDEV, No such device"
69 76 65 64 20 68 63 75 73 20 6F 4E     0418
                                  65 63 0424
                                     19 040C
2C 45 4C 49 46 48 43 55 53 4F 4E 00'    0426    346 20$:     .ASCIC  "NOSUCHFILE, No such file"
6C 69 66 20 68 63 75 73 20 6F 4E 20     0432
                                  65 043E
                                     18 0426
6E 20 65 6C 69 66 20 2C 46 4E 46 00'    043F    347 30$:     .ASCIC  "FNF, file not found"
            64 6E 75 6F 66 20 74 6F     C44B
                                     13 043F
65 64 20 64 61 62 20 2C 56 45 44 00'    0453    348 40$:     .ASCIC  "DEV, bad device, or inappropriate device type"
61 6E 69 20 72 6F 20 2C 65 63 69 76     045F
64 20 65 74 61 69 72 70 6F 72 70 70     046B
      65 70 79 74 20 65 63 69 76 65     0477
                                     2D 0453
74 63 65 72 69 64 20 2C 46 4E 44 00'    0481    349 50$:     .ASCIC  "DNF, directory not found"
6E 75 6F 66 20 74 6F 6E 20 79 72 6F     048D
                                  64 0499
                                     18 0481
73 73 65 63 63 61 20 2C 43 43 41 00'    049A    350 60$:     .ASCIC  "ACC, access violation"
      6E 6F 69 74 61 6C 6F 69 76 20     04A6
                                     15 049A
66 72 65 76 6F 20 64 6C 65 69 46 00'    04B0    351 70$:     .ASCIC  "Field overflow"
                                  77 6F 6C 04BC
                                     0E 04B0
20 74 75 70 6E 69 20 6C 6C 75 4E 00'    04BF    352 80$:     .ASCIC  "Null input string"
                     67 6E 69 72 74 73  04CB
                                     11 04BF
61 68 63 20 6C 61 67 65 6C 6C 49 00'    04D1    353 90$:     .ASCIC  "Illegal character"
                     72 65 74 63 61 72  04DD
                                     11 04D1
20 72 65 6D 6D 61 72 67 6F 72 50 00'    04E3    354 100$:    .ASCIC  "Programmer error"
                           72 6F 72 72 65  04EF
                                     10 04E3
61 63 6E 75 72 74 20 61 74 61 44 00'    04F4    355 110$:    .ASCIC  "Data truncated"
                                  64 65 74 0500
                                     0E 04F4
63 65 76 20 64 69 6C 61 76 6E 49 00'    0503    356 120$:    .ASCIC  "Invalid vector"
                                  72 6F 74 050F
                                     0E 0503
63 65 76 20 64 69 6C 61 76 6E 49 00'    0512    357 130$:    .ASCIC  "Invalid vector address"
   73 73 65 72 64 64 61 20 72 6F 74     051E
                                     16 0512
64 61 20 6C 61 72 75 74 72 69 56 00'    0529    358 140$:    .ASCIC  "Virtural address space full"
20 65 63 61 70 73 20 73 73 65 72 64     0535
                           6C 6C 75 66  0541
                                     1B 0529
6E 65 69 63 69 66 66 75 73 6E 49 00'    0545    359 150$:    .ASCIC  "Insufficient memory"
            79 72 6F 6D 65 6D 20 74     0551
                                     13 0545
61 68 63 20 6C 61 69 74 69 6E 49 00'    0559    360 160$:    .ASCIC  "Initial channel hardware error"
```

```
72 61 77 64 72 61 68 20 6C 65 6E 6E  0565
                72 6F 72 72 65 20 65  0571
                                  1E  0559
6E 6E 61 68 63 20 6C 61 6E 69 46 00'  0578    361 170$:   .ASCIC  "Final channel hardware error"
20 65 72 61 77 64 72 61 68 20 6C 65   0584
                   72 6F 72 72 65     0590
                                  1C  0578
67 6F 6C 20 6D 61 72 67 6F 72 50 00'  0595    362 180$:   .ASCIC  "Program logic error"
                72 6F 72 72 65 20 63 69  05A1
                                  13  0595
67 61 70 20 6C 61 67 65 6C 6C 49 00'  05A9    363 190$:   .ASCIC  "Illegal page count"
                74 6E 75 6F 63 20 65  05B5
                                  12  05A9
66 20 73 69 20 72 65 66 66 75 42 00'  05BC    364 200$:    ASCIC  "Buffer is fragmented"
          64 65 74 6E 65 6D 67 61 72  C5C8
                                  14  05BC
65 68 63 20 65 6E 69 68 63 61 4D 00'  05D1    365 210$:   .ASCIC  "Machine check"
                                  6B 63  05DD
                                  0D  05D1
63 61 74 73 20 6C 65 6E 72 65 4B 00'  05DF    366 220$:   .ASCIC  "Kernel stack not valid"
   64 69 6C 61 76 20 74 6F 6E 20 6B   05EB
                                  16  05DF
20 6C 69 61 66 20 72 65 77 6F 50 00'  05F6    367 230$:   .ASCIC  "Power fail interrupt"
          74 70 75 72 72 65 74 6E 69  0602
                                  14  05F6
6E 6F 69 74 61 6C 73 6E 61 72 54 00'  060B    368 240$:   .ASCIC  "Translation not valid"
      64 69 6C 61 76 20 74 6F 6E 20   0617
                                  15  060B
65 6D 65 6C 70 6D 69 20 74 6F 4E 00'  0621    369 250$:   .ASCIC  "Not implemented"
                   64 65 74 6E 65     062D
                                  0F  0621
67 69 68 20 6F 6F 74 20 4C 50 49 00'  0631    370 260$:   .ASCIC  "IPL too high"
                                  68 63  063D
                                  0C  0631
6C 63 20 6C 61 76 72 65 74 6E 49 00'  063E    371 270$:   .ASCIC  "Interval clock error"
          72 6F 72 72 65 20 6B 63 6F  064A
                                  14  063E
20 63 69 74 65 6D 68 74 69 72 41 00'  0653    372 280$:   .ASCIC  "Arithmetic trap"
                   70 61 72 74 65     065F
                                  0F  0653
20 64 65 74 63 65 70 78 65 6E 55 00'  0663    373 290$:   .ASCIC  "Unexpected interrupt"
          74 70 75 72 72 65 74 6E 69  066F
                                  14  0663
69 66 20 50 43 41 20 2C 43 43 41 00'  0678    374 300$:   .ASCIC  "ACC, ACP file access failed"
61 66 20 73 73 65 63 63 61 20 65 6C   0684
                   64 65 6C 69 6C     0690
                                  1B  0678
74 6F 6E 6E 61 63 20 2C 52 43 43 00'  0694    375 310$:   .ASCIC  "CCR, cannot connect RAB"
42 41 52 20 74 63 65 6E 6E 6F 63 20   06A0
                                  17  0694
69 6D 61 6E 79 64 20 2C 45 4D 44 00'  06AC    376 320$:   .ASCIC  "DME, dynamic memory exhausted"
68 78 65 20 79 72 6F 6D 65 6D 20 63   06B8
                   64 65 74 73 75 61  06C4
                                  1D  06AC
66 6F 20 64 6E 65 20 2C 46 4F 45 00'  06CA    377 330$:   .ASCIC  "EOF, end of file detected"
74 63 65 74 65 64 20 65 6C 69 66 20   06D6
                                  64 65  06E2
```

```
                                                        19  06CA
69 6C 61 76 6E 69 20 2C 42 41 46 00'  06E4    378 340$:   .ASCIC  "FAB, invalid FAB or FAB not accessible"
42 41 46 20 72 6F 20 42 41 46 20 64   06F0
69 73 73 65 63 63 61 20 74 6F 6E 20   06FC
                        65 6C 62      0708
                                 26   06E4
69 6C 61 76 6E 69 20 2C 49 46 49 00'  070B    379 350$:   .ASCIC  "IFI, invalid internal file identifier (IFI) value"
66 20 6C 61 6E 72 65 74 6E 69 20 64   0717
69 66 69 74 6E 65 64 69 20 65 6C 69   0723
6C 61 76 20 29 49 46 49 28 20 72 65   072F
                        65 75 73 74   073B
                                 31   070B
74 61 72 65 70 6F 20 2C 50 4F 49 00'  073D    380 360$:   .ASCIC  "IOP, operation invalid for file organization or device"
20 64 69 6C 61 76 6E 69 20 6E 6F 69   0749
67 72 6F 20 65 6C 69 66 20 72 6F 66   C755
72 6F 20 6E 6F 69 74 61 7A 69 6E 61   0761
                  65 63 69 76 65 64   076D
                                 36   073D
69 6C 61 76 6E 69 20 2C 49 53 49 00'  0774    381 370$:   .ASCIC  "ISI, invalid internal stream identifier (ISI) value"
73 20 6C 61 6E 72 65 74 6E 69 20 64   0780
69 74 6E 65 64 69 20 6D 61 65 72 74   078C
76 20 29 49 53 49 28 20 72 65 69 66   0798
                        65 75 6C 61   07A4
                                 33   0774
69 6C 61 76 6E 69 20 2C 42 41 52 00'  07A8    382 380$:   .ASCIC  "RAB, invalid RAB or RAB not accessible"
42 41 52 20 72 6F 20 42 41 52 20 64   07B4
69 73 73 65 63 63 61 20 74 6F 6E 20   07C0
                        65 6C 62      07CC
                                 26   07A8
72 20 65 6C 69 66 20 2C 52 45 52 00'  07CF    383 390$:   .ASCIC  "RER, file read error"
         72 6F 72 72 65 20 64 61 65   07DB
                                 14   07CF
64 72 6F 63 65 72 20 2C 42 54 52 00'  07E4    384 400$:   .ASCIC  "RTB, record too large for user's buffer"
66 20 65 67 72 61 6C 20 6F 6F 74 20   07F0
75 62 20 73 27 72 65 73 75 20 72 6F   07FC
                        72 65 66 66   0808
                                 27   07E4
69 6C 61 76 6E 69 20 2C 41 46 52 00'  080C    385 410$:   .ASCIC  "RFA, invalid record's file address (RFA)"
66 20 73 27 64 72 6F 63 65 72 20 64   0818
20 73 73 65 72 64 64 61 20 65 6C 69   0824
                     29 41 46 52 28   0830
                                 28   080C
61 67 65 6C 6C 69 20 2C 43 52 49 00'  0835    386 420$:   .ASCIC  "IRC, illegal record encountered in file"
63 6E 65 20 64 72 6F 63 65 72 20 6C   0841
20 6E 69 20 64 65 72 65 74 6E 75 6F   084D
                        65 6C 69 66   0859
                                 27   0835
20 72 6F 72 72 65 20 2C 4D 4E 46 00'  085D    387 430$:   .ASCIC  "FNM, error in filename"
   65 6D 61 6E 65 6C 69 66 20 6E 69   0869
                                 16   085D
61 66 20 2C 52 52 45 4C 52 54 43 00'  0874    388 440$:   .ascic  "CTRLERR, fatal controller error"       ;                    [12]
6C 6C 6F 72 74 6E 6F 63 20 6C 61 74   0880
            72 6F 72 72 65 20 72 65   088C
                                 1F   0874
                                      0894    389
                                      0894    390 SEVERITY:
                        3F 46 45 57   0894    391         .ASCII  "WEF?"
```

```
                     0898   393          .SBTTL  DSX$ErrSoft Routine
                 00000000   394          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   395 ;++
                     0000   396 ; FUNCTIONAL DESCRIPTION:
                     0000   397 ;
                     0000   398 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
                     0000   399 ;       ''ERRSOFT'' CALL TO THE OPERATOR AND A.P.T..
                     0000   400 ;
                     0000   401 ; CALLING SEQUENCE:
                     0000   402 ;
                     0000   403 ;       NONE
                     0000   404 ;
                     0000   405 ; INPUT PARAMETERS:
                     0000   406 ;
                     C000   407 ;       ERR$_NUM(AP)     = ERROR NUMBER
                     0000   408 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
                     0000   409 ;       ERR$_MSGADR(AP) = POINTER TO ASCII STRING
                     0000   410 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
                     0000   411 ;
                     0000   412 ; IMPLICIT INPUTS:
                     0000   413 ;
                     0000   414 ;       NONE
                     0000   415 ;
                     0000   416 ; OUTPUT PARAMETERS:
                     0000   417 ;
                     0000   418 ;       NONE
                     0000   419 ;
                     0000   420 ; IMPLICIT OUTPUTS:
                     0000   421 ;
                     0000   422 ;       NONE
                     0000   423 ;
                     0000   424 ; COMPLETION CODES:
                     0000   425 ;
                     0000   426 ;       NONE
                     0000   427 ;
                     0000   428 ; SIDE EFFECTS:
                     0000   429 ;
                     0000   430 ;       NONE
                     0000   431 ;
                     0000   432 ;--
```

```
                      0FFC  0000   434 .ENTRY  DSX$ERRSOFT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                [14]
                            0002   435
          00000000'EF   D6  0002   436         Incl    L^Ds$GL_SoftErr_Count   ; Count this soft error          [15]
          0000000D'EF   DF  0008   437         PUSHAL  L^SOFTER                ; INDICATE ERROR TYPE            [08]
                00C0    30  000E   438         BSBW    RRPTERROR               ; CALL ERROR REPORTING SUBRTN
     0000FE40'EF  04    D0  0011   439         MOVL    #APM$_SOFTERR, -        ; Indicate error type in mailbox
                            0018   440                   L^DSA$GL_MSGTYP
                            0018   441         Br_If_Not_APT   10$             ; If not running from APT, branch  [19]
          00000000'EF   16  0020   442         JSB     L^APT_MSG               ; Else, transfer control to APT  [08]
                            0026   443
                      04    0026   444 10$:    RET
                            0027   445         .SBTTL  DSX$ErrHard Routine
                            0027   446 ;++
                            0027   447 ; FUNCTIONAL DESCRIPTION:
                            C027   448 ;
                            0027   449 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
                            0027   450 ;       "ERRHARD" CALL TO THE OPERATOR AND A.P.T..
                            0027   451 ;
                            0027   452 ; CALLING SEQUENCE:
                            0027   453 ;
                            0027   454 ;       NONE
                            0027   455 ;
                            0027   456 ; INPUT PARAMETERS:
                            0027   457 ;
                            0027   458 ;       ERR$_NUM(AP)     = ERROR NUMBER
                            0027   459 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
                            0027   460 ;       ERR$_MSGADR(AP)  = POINTER TO ASCII STRING
                            0027   461 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
                            0027   462 ;
                            0027   463 ; IMPLICIT INPUTS:
                            0027   464 ;
                            0027   465 ;       NONE
                            0027   466 ;
                            0027   467 ; OUTPUT PARAMETERS:
                            0027   468 ;
                            0027   469 ;       NONE
                            0027   470 ;
                            0027   471 ; IMPLICIT OUTPUTS:
                            0027   472 ;
                            0027   473 ;       NONE
                            0027   474 ;
                            0027   475 ; COMPLETION CODES:
                            0027   476 ;
                            0027   477 ;       NONE
                            0027   478 ;
                            0027   479 ; SIDE EFFECTS:
                            0027   480 ;
                            0027   481 ;       NONE
                            0027   482 ;
                            0027   483 ;--
```

```
                     0FFC  0027   485 .ENTRY  DSX$ERRHARD,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                    [14]
                           0029   486
          00000000'EF   D6  0029   487         Incl    L^Ds$GL_HardErr_Count   ; Count this hard error             [15]
          00000001'EF   DF  002F   488         PUSHAL  L^HARDER                ; INDICATE ERROR TYPE               [08]
                0099    30  0035   489         BSBW    RRPTERROR               ; CALL ERROR REPORTING SUBRTN
    0000FE40'EF   03    D0  0038   490         MOVL    #APM$_HARDERR, -        ; Indicate error type in mailbox
                           003F   491                   L^DSA$GL_MSGTYP
                           003F   492         Br_If_Not_APT   10$              ; If not running from APT, branch   [19]
          00000000'EF   16  0047   493         JSB     L^APT_MSG               ; Else, transfer control to APT     [08]
                           004D   494
                     04    004D   495 10$:    RET
```

```
                     004E    497          .SBTTL  DSX$ErrPrep Routine                                                [17]
                     004E    498 ;++                                                                                 [17]
                     004E    499 ; FUNCTIONAL DESCRIPTION:                                                           [17]
                     004E    500 ;                                                                                   [17]
                     004E    501 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN                           [17]
                     004E    502 ;       ''ERRPREP'' CALL TO THE OPERATOR AND A.P.T..                                [17]
                     004E    503 ;                                                                                   [17]
                     004E    504 ; CALLING SEQUENCE:                                                                 [17]
                     004E    505 ;                                                                                   [17]
                     004E    506 ;       NONE                                                                        [17]
                     004E    507 ;                                                                                   [17]
                     004E    508 ; INPUT PARAMETERS:                                                                 [17]
                     004E    509 ;                                                                                   [17]
                     004E    510 ;       ERR$_NUM(AP)     = ERROR NUMBER                                             [17]
                     004E    511 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER                                      [17]
                     004E    512 ;       ERR$_MSGADR(AP) = POINTER TO ASCII STRING                                   [17]
                     004E    513 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE                                     [17]
                     004E    514 ;                                                                                   [17]
                     004E    515 ; IMPLICIT INPUTS:                                                                  [17]
                     004E    516 ;                                                                                   [17]
                     004E    517 ;       NONE                                                                        [17]
                     004E    518 ;                                                                                   [17]
                     004E    519 ; OUTPUT PARAMETERS:                                                                [17]
                     004E    520 ;                                                                                   [17]
                     004E    521 ;       NONE                                                                        [17]
                     004E    522 ;                                                                                   [17]
                     004E    523 ; IMPLICIT OUTPUTS:                                                                 [17]
                     004E    524 ;                                                                                   [17]
                     004E    525 ;       NONE                                                                        [17]
                     004E    526 ;                                                                                   [17]
                     004E    527 ; COMPLETION CODES:                                                                 [17]
                     004E    528 ;                                                                                   [17]
                     004E    529 ;       NONE                                                                        [17]
                     004E    530 ;                                                                                   [17]
                     004E    531 ; SIDE EFFECTS:                                                                     [17]
                     004E    532 ;                                                                                   [17]
                     004E    533 ;       NONE                                                                        [17]
                     004E    534 ;                                                                                   [17]
                     004E    535 ;--                                                                                 [17]
```

```
                         0FFC  004E   537 .ENTRY  DSX$ERRPREP,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                          [17]
                               0050   538
            00000000'EF    D6  0050   539         Incl    L^Ds$GL_PrepErr_Count   ; Count this device preparation  error  [17]
            0000002D'EF    DF  0056   540         PUSHAL  L^PREPER                ; INDICATE ERROR TYPE                  [17]
                    0072   30  005C   541         BSBW    RRPTERROR               ; CALL ERROR REPORTING SUBRTN          [17]
      0000FE40'EF    0C    D0  005F   542         MOVL    #APM$_PREPERR, -        ; Indicate error type in mailbox       [17]
                               0066   543                   L^DSA$GL_MSGTYP
                               0066   544         Br_If_Not_APT   10$             ; If not running from APT, branch      [19]
            00000000'EF    16  006E   545         JSB     L^APT_MSG               ; Else, transfer control to APT        [17]
                               0074   546
                         04    0074   547 10$:    RET
```

```
0075    549          .SBTTL  DSX$ErrDev Routine
0075    550 ;++
0075    551 ; FUNCTIONAL DESCRIPTION:
0075    552 ;
0075    553 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
0075    554 ;       "ERRDEV" CALL TO THE OPERATOR AND A.P.T..
0075    555 ;
0075    556 ; CALLING SEQUENCE:
0075    557 ;
0075    558 ;       NONE
0075    559 ;
0075    560 ; INPUT PARAMETERS:
0075    561 ;
0075    562 ;       ERR$_NUM(AP)     = ERROR NUMBER
0075    563 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
0075    564 ;       ERR$_MSGADR(AP) = POINTER TO ASCII STRING
0075    565 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
0075    566 ;
0075    567 ; IMPLICIT INPUTS:
0075    568 ;
0075    569 ;       NONE
0075    570 ;
0075    571 ; OUTPUT PARAMETERS:
0075    572 ;
0075    573 ;       NONE
0075    574 ;
0075    575 ; IMPLICIT OUTPUTS:
0075    576 ;
0075    577 ;       NONE
0075    578 ;
0075    579 ; COMPLETION CODES:
0075    580 ;
0075    581 ;       NONE
0075    582 ;
0075    583 ; SIDE EFFECTS:
0075    584 ;
0075    585 ;       NONE
0075    586 ;
0075    587 ;--
```

```
                          0FFC  0075   589 .ENTRY  DSX$ERRDEV,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>  ;                    [14]
                                0077   590
            00000000'EF    D6  0077   591         Incl    L^Ds$GL_DevErr_Count    ; Count this device error            [15]
            00000019'EF    DF  007D   592         PUSHAL  L^DEVFER                ; INDICATE ERROR TYPE                [08]
      00000000'EF    09    E2  0083   593         BBSS    #DS$V_DEVFLG, -         ; FLAG DEVICE-FATAL                  [08]
                     00        008A   594                   L^DS$GL_FLAGS,10$
                               008B   595
                     44    10  008B   596 10$:    BSBB    RRPTERROR               ; CALL ERROR REPORTING SUBRTN        [10]
      0000FE40'EF    02    D0  008D   597         MOVL    #APM$_DEVERR, -         ; Indicate error type in mailbox
                               0094   598                   L^DSA$GL_MSGTYP
                               0094   599         Br_If_Not_APT   20$             ; If not running from APT, branch    [19]
            00000000'EF    16  009C   600         JSB     L^APT_MSG               ; Else, transfer control to APT      [08]
                               00A2   601
                     04        00A2   602 20$:    RET
```

```
                        00A3   604          .SBTTL  DSX$ErrSys Routine
                        00A3   605 ;++
                        00A3   606 ; FUNCTIONAL DESCRIPTION:
                        00A3   607 ;
                        00A3   608 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
                        00A3   609 ;       "ERRSYS" CALL TO THE OPERATOR AND A.P.T..
                        00A3   610 ;
                        00A3   611 ; CALLING SEQUENCE:
                        00A3   612 ;
                        00A3   613 ;       NONE
                        00A3   614 ;
                        00A3   615 ; INPUT PARAMETERS:
                        00A3   616 ;
                        00A3   617 ;       ERR$_NUM(AP)     = ERROR NUMBER
                        C0A3   618 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
                        00A3   619 ;       ERR$_MSGADR(AP) = POINTER TO ASCII STRING
                        00A3   620 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
                        00A3   621 ;
                        00A3   622 ; IMPLICIT INPUTS:
                        00A3   623 ;
                        00A3   624 ;       NONE
                        00A3   625 ;
                        00A3   626 ; OUTPUT PARAMETERS:
                        00A3   627 ;
                        00A3   628 ;       NONE
                        00A3   629 ;
                        00A3   630 ; IMPLICIT OUTPUTS:
                        00A3   631 ;
                        00A3   632 ;       NONE
                        00A3   633 ;
                        00A3   634 ; COMPLETION CODES:
                        00A3   635 ;
                        00A3   636 ;       NONE
                        00A3   637 ;
                        00A3   638 ; SIDE EFFECTS:
                        00A3   639 ;
                        00A3   640 ;       NONE
                        00A3   641 ;
                        00A3   642 ;--
```

```
                    OFFC  00A3   644 .ENTRY  DSX$ERRSYS,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>  ;                    [14]
                          00A5   645
      00000000'EF     D6  00A5   646         Incl    L^Ds$GL_SysErr_Count    ; Count this system error            [15]
      00000048'EF     DF  00AB   647         PUSHAL  L^SYSFER                ; INDICATE ERROR TYPE                [08]
00000000'EF   0A      E2  00B1   648         BBSS    #DS$V_SYSFLG, -         ; FLAG SYSTEM-FATAL                  [08]
              00            00B8   649                   L^D5$GL_FLAGS, 10$
                          00B9   650
              16      10  00B9   651 10$:     BSBB    RRPTERROR               ; CALL ERROR REPORTING SUBRTN        [10]
0000FE40'EF   01      DC  00BB   652         MOVL    #APM$_SYSERR, -         ; Indicate error type in mailbox
                          00C2   653                   L^DSA$GL_MSGTYP
                          00C2   654         Br_If_Not_APT   20$             ; If not running from APT, branch    [19]
      00000000'EF     16  00CA   655         JSB     L^APT_MSG               ; Else, transfer control to APT      [08]
                          00D0   656
                      04  00D0   657 20$:     RET
```

```
                        00D1    659          .SBTTL  RRtpError Routine
                        00D1    660 ;++
                        00D1    661 ; FUNCTIONAL DESCRIPTION:
                        00D1    662 ;
                        00D1    663 ;       THIS ROUTINE REPORTS THE ERROR MESSAGE TO THE OPERATOR,
                        00D1    664 ;       IT IS REACHED VIA "BSBW" FROM A SUPERVISOR ROUTINE.
                        00D1    665 ;
                        00D1    666 ; CALLING SEQUENCE:
                        00D1    667 ;
                        00D1    668 ;       NONE
                        00D1    669 ;
                        00D1    670 ; INPUT PARAMETERS:
                        00D1    671 ;
                        00D1    672 ;       4(SP)             = POINTER TO ERROR TYPE STRING
                        00D1    673 ;       ERR$_NUM(AP)      = ERROR NUMBER
                        00D1    674 ;       ERR$_UNIT(AP)     = LOGICAL UNIT NUMBER
                        00D1    675 ;       ERR$_MSGADR(AP)   = POINTER TO ASCII STRING
                        00D1    676 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
                        00D1    677 ;
                        00D1    678 ; IMPLICIT INPUTS:
                        00D1    679 ;
                        00D1    680 ;       NONE
                        00D1    681 ;
                        00D1    682 ; OUTPUT PARAMETERS:
                        00D1    683 ;
                        00D1    684 ;       NONE
                        00D1    685 ;
                        00D1    686 ; IMPLICIT OUTPUTS:
                        00D1    687 ;
                        00D1    688 ;       NONE
                        00D1    689 ;
                        00D1    690 ; COMPLETION CODES:
                        00D1    691 ;
                        00D1    692 ;       NORMAL RETURN IF "HALT" ON ERROR IS NOT SET
                        00D1    693 ;       GO TO COMMAND MODE IF "HALT" IS SET (IMMEDIATELY AFTER RETURN)
                        00D1    694 ;
                        00D1    695 ; SIDE EFFECTS:
                        00D1    696 ;
                        00D1    697 ;       NONE
                        00D1    698 ;
                        00D1    699 ;--
```

```
                                00D1   701 RRPTERROR:                                                  ;
                00000000'EF     D6  00D1   702          INCL    L^DS$GL_ERRCNT                          ; KEEP TOTAL OF ERRORS          [08]
                                00D7   703          Set_ErrFlg                                      ; Set the error flag            [19]
                00000000'EF     D4  00DF   704          CLRL    L^DS$GA_CHKLPPC                         ; CLEAR THE CHECK_LOOP P.C.     [08]
                0000FE68'EF     7C  00E5   705          CLRQ    L^DSA$GQ_MSGPTR                         ; Clear message descriptor
          0000FE44'EF   04 AC   3C  00EB   706          MOVZWL  ERR$_NUM(AP), L^DSA$GL_ERRNO            ; Save error number-only a word [16]
                                00F3   707          Br_If_Not_Bell  20$                             ; Skip if "BELL" flag not set   [19]
                                00FB   708          $QIOW_S ,[^DS$GW_TTOUT, -                       ; "DING"                        [08]
                                00FB   709                  #IO$_WRITEVBLK,,, -
                                00FB   710                  L^MSGBELL, #1
                                0120   711
                                0120   712 20$:     $DS_GPHARD_S  ERR$_UNIT(AP), -(SP)              ; GET ADDRESS OF UUT'S P-TABLE
                      51   8E   D0  012C   713          MOVL    (SP)+, R1                               ;  INTO R1
                         13 50   E9  012F   714          BLBC    R0, 30$                                 ; IF NON-EXISTENT, FAKE IT
          00000000'EF   61   01   C3  C132   715          SUBL3   #1, HP$Q_DEVICE(R1), L^Q_BUFQWD         ; REDUCE COUNT BY ONE           [08]
           00000004'EF   04 A1   01   C1  013A   716          ADDL3   #1, HP$Q_DEVICE+4(R1), -                ; ADJUST POINTER (SKIP "_")     [08]
                                0143   717                  L^Q_BUFQWD+4                            ;                               [08]
                            08  11  0143   718          BRB     40$                                     ; AND PROCEED
                                0145   719
          00000000'EF       0000075'EF   7D  0145   720 30$:     MOVQ    L^QUNKDEV, L^Q_BUFQWD           ; ELSE USE "UNK" MSG            [08]
                                0150   721
          0000FE58'EF       00000000'EF  3C  0150   722 40$:     MOVZWL  L^Q_BUFQWD,L^DSA$GL_DEVLEN      ; Dev name string len to mailbox[08]
                                07  BB  0158   723          PUSHR   #^M<R0,R1,R2>                   ; CLOBBERED BY "MOVC3"
                      50        00000000'EF  7D  015D   724          MOVQ    Q_BUFQWD,R0             ; Get length/address of name
                         50     50   3C  0164   725          movzwl  r0, r0                          ; Zero extend length            [10]
                      52        0000FE5C'EF  9E  0167   726          MOVAB   L^DSA$GT_DEVNAM,R2
                                016E   727
                         82   81   90  016E   728 45$:     MOVB    (R1)+,(R2)+
                            FA 50   F5  0171   729          SOBGTR  R0,45$
                                07  BA  0174   730          POPR    #^M<R0,R1,R2>                           ; RESTORE CLOBBERED REGISTERS
```

```
                                0176   732          Br_If_IE1       100$                    ; Inhibit level-1 flag set?
                                017E   733                                                  ;
                        00F3    30     017E   734 50$:     BSBW    DS_TESTID               ; GET TEST/SUBTEST MESSAGE.
                   50   0C BC   DE     0181   735          MOVAL   @ERR$_MSGADR(AP),R0     ; GET ADDRESS OF CALLER'S MSG
                        07      12     0185   736          BNEQ    90$                     ; PROCEED IF SPECIFIED
                50 00000074'EF  DE     0187   737          MOVAL   L^ANULL,R0              ; Else, use a null string               [08]
                                018E   738
                   51   FC BD   9A     018E   739 90$:     movzbl  @-4(fp), r1             ; Get error type code                   [10]
                        FC AD   D6     0192   740          incl    -4(fp)                  ; And increment address past type       [10]
                                0195   741
                                0195   742          ;+
                                0195   743          ; Now, check to see if CRD is running. If so, load the address of the   [22]
                                0195   744          ; user's error text into the User_Error_Text location for use by CRD.   [22]
                                0195   745          ;-
                                C195   746
                        01      BB     0195   747          PushR   #^M<R0>                 ; Save R0 because it's clobbered        [22]
                00000000'EF     16     0197   748          Jsb     L^DSR$Check_MenuTest_Off_Set; Check if Menu Test  Offline bit is set
                                019D   749                                                  ;                                       [23]
                     09 50      E8     019D   750          Blbs    R0, 93$                 ; If Menu Test running, branch          [22]
                00000000'EF     16     01A0   751          Jsb     L^DSR$Check_AutoTest_Off_Set; Check if Auto Test Offlinebit is set
                                01A6   752                                                  ;                                       [23]
                     0B 50      E9     01A6   753          Blbc    R0, 95$                 ; No Auto Test, no Menu Test => branch  [22]
                                01A9   754
                        01      BA     01A9   755 93$:     PopR    #^M<R0>                 ; Restore R0 - MsgAdr parameter         [22]
                00000000'EF     50     01AB   756          MovL    R0, -                   ; Move the MsgAdr parameter into a      [22]
                                01B2   757                    L^DS$GA_User_Error_Text   ; ... location for CRD.                 [20]
                        02      11     01B2   758          Brb     96$                     ; Go print it                           [22]
                                01B4   759
                        01      BA     01B4   760 95$:     PopR    #^M<R0>                 ; Restore R0                            [22]
                                01B6   761
                                01B6   762 96$:     $PRINT  r1, -                   ; Print the error message:              [22]
                                01B6   763                    #DS$K_PrintF, -           ; ... printf                            [22]
                                01B6   764                    L^FMTERRHDR, -            ; ... format string                     [22]
                                01B6   765                    @#L$A_NAME, -             ; ... diag name                         [22]
                                01B6   766                    @#L$L_REV, -              ; ... diag rev level                    [22]
                                01B6   767                    @#L$L_UPDATE, -           ; ... diag update level                 [22]
                                01B6   768                    L^DSA$GL_PASSNO, -        ; ... pass number of the diagnostic     [22]
                                01B6   769                    #DS$GQ_TESTID, -          ; ... ???                               [22]
                                01B6   770                    L^DSA$GL_ERRNO, -         ; ... error number                      [22]
                                01B6   771                    #0, -                     ; LINE 2
                                01B6   772                    -4(FP), -                 ;
                                01B6   773                    #Q_BUFQWD, -              ; LINE 3
                                01B6   774                    R0                        ; ... MsgAdr - user's error text        [22]
                                01FA   775
                     10 AC      D5     01FA   776 100$:     TSTL    ERR$_POINTER(AP)        ; IS THERE A PRINT LINK ?
                        08      13     01FD   777          BEQL    110$                    ; NO, SKIP
                00000000'EF     09     90     01FF   778          movb    #ds$k_type_error_body, -; Set type code to error body   [10]
                                0206   779                    L^ds$gb_typecode        ;                                       [10]
                   10 BC        6C     FA     0206   780          CALLG   (AP), @ERR$_POINTER(AP) ; CALL USER'S PRINT ROUTINE
                                020A   781
                                020A   782 110$:     Br_If_IE1       115$                    ; Don't print trailer if IE1            [19]
                                0212   783          $print  #ds$k_type_error_end, -  ; Print out end-of-error message       [10]
                                0212   784                    #ds$k_printf, -           ; .. use PRINTF to type it              [10]
                                0212   785                    fmterror, -               ; .. 'end-of-error' format              [10]
                                0212   786                    -4(fp), -                 ; .. Type of error                      [10]
                                0212   787                    L^dsa$gl_errno            ; .. and error number                   [10]
                                022E   788
```

```
                                022E    789 115$:   Br_If_Halt      120$            ; Halt on error detection?                [19]
                          32    11 0236   790         BRB     RRPTERRORX          ; NO HALT(S), RETURN
                                0238    791                                         ;
                                0238    792 120$:   Set_Halt                        ; YES HALT, SET INTERNAL FLAG             [19]
00000000'E[     10 BD     DE    0240    793         MOVAL   @CF$L_PC(FP), -         ; SAVE RETURN 'PC'                        [08]
                                0248    794           L^DS$AA_BPTADDR
00000000'EF     10 BD     90    0248    795         MOVB    @CF$L_PC(FP), -         ; SAVE THE OPCODE LOCATED THERE           [08]
                                0250    796           L^DS$AB_BPTINST
                10 BD     03    90 0250   797         MOVB    #BPTCODE,@CF$L_PC(FP)   ; CAUSE A BREAK TO THE "CLI"
                                0254    798         $print  #ds$k_type_err_halt, -  ; Print out HALT ON ERROR message          [10]
                                0254    799           #ds$k_printf, -               ; .. Use Printf to type it                [10]
                                0254    800           l^fmterrhlt, -                ; .. 'Halt on error' message format       [10]
                                0254    801           cf$l_pc(fp)                   ; .. display "halt pc"                    [10]
                                026A    802                                         ;
                                C26A    803 RRPTERRORX:                             ;
                                026A    804         QA_MAIN Error_Error_End         ;                                         [07]
                          05    0273    805         RSB                             ; RETURN TO THE DIAGNOSTIC
```

```
                                0274   807          .SBTTL  DS_TestID Routine
                                0274   808 ;+
                                0274   809 ; FUNCTIONAL DESCRIPTION:
                                0274   810 ;
                                0274   811 ;       DETERMINES IF IN INIT OR CLEANUP CODE AND FORMS THE PROPER MESSAGE.
                                0274   812 ;
                                0274   813 ;--
                                0274   814 DS_TESTID::
            0000FE50'EF   D5    0274   815          TSTL    L^DSA$GL_TESTNO                 ; Test 0 ?
                          2C    12  027A   816          BNEQ    70$                     ; NO, PROCEED WITH STANDARD MSG
                                027C   817                                              ;
         01 0000FE4C'EF   D1    027C   818          CMPL    L^DSA$GL_SUBTNO,#1              ; Test 0 & subtest 1 ?
                          0D    12  0283   819          BNEQ    60$                     ; NO, TRY NEXT
00000008'EF 0000008B'EF   7D    0285   820          MOVQ    L^QINITSEC,L^DS$GQ_TESTID       ; Yes, use init section msg     [08]
                          4D    11  0290   821          BRB     80$                     ;  AND PROCEED
         02 0000FE4C'EF   D1    0292   822 60$:     CMPL    L^DSA$GL_SUBTNO,#2              ; Test 0 & subtest 2 ?
                          0D    12  0299   823          BNEQ    70$                     ; NO, TRY NEXT
00000008'EF 000000A9'EF   7D    029B   824          MOVQ    L^QCLEANSEC,L^DS$GQ_TESTID      ; Yes, use clean section msg    [08]
                          37    11  02A6   825          BRB     80$                     ;  AND PROCEED
   00000008'EF    20    3C    02A8   826 70$:     MOVZWL  #TESTID_LEN,L^DS$GQ_TESTID      ; SET UP QUADWORD DESCRIPTOR    [08]
0000000C'EF 00000010'EF   DE    02AF   827          MOVAL   L^T_TESTID,L^DS$GQ_TESTID+4     ;  FOR ''TEST/SUBTEST'' STRING  [08]
                                02BA   828          $FAO_S  L^QFMTTESTID,L^DS$GQ_TESTID,-   ; Convert test and subtest     [08]
                                02BA   829                  L^DS$GQ_TESTID,-            ;   numbers                     [08]
                                02BA   830                  L^DSA$GL_TESTNO,L^DSA$GL_SUBTNO ;  into ascii for output
                          05    02DF   831 80$:     RSB                             ; RETURN.
```

```
                     833          .SBTTL  DS_ErrSup Routine
02E0                 834 ;++
02E0                 835 ; FUNCTIONAL DESCRIPTION:
02E0                 836 ;
02E0                 837 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
02E0                 838 ;       "ERRSUP" CALL TO THE OPERATOR AND A.P.T..
02E0                 839 ;
02E0                 840 ; CALLING SEQUENCE:
02E0                 841 ;
02E0                 842 ;       CALLG   ARGLIST, @#DS_ERRSUP
02E0                 843 ;         OR
02E0                 844 ;       CALLS   #3, @#DS_ERRSUP
02E0                 845 ;
02E0                 846 ; INPUT PARAMETERS:
02E0                 847 ;
02E0                 848 ;        4(AP) ERROR NUMBER
02E0                 849 ;        8(AP) POINTER TO COUNTED ASCII ERROR MESSAGE STRING
02E0                 850 ;       12(AP) POINTER TO COUNTED ASCII MODULE NAME STRING
02E0                 851 ;
02E0                 852 ; IMPLICIT INPUTS:
02E0                 853 ;
02E0                 854 ;       NONE
02E0                 855 ;
02E0                 856 ; OUTPUT PARAMETERS:
02E0                 857 ;
02E0                 858 ;       NONE
02E0                 859 ;
02E0                 860 ; IMPLICIT OUTPUTS:
02E0                 861 ;
02E0                 862 ;       NONE
02E0                 863 ;
02E0                 864 ; COMPLETION CODES:
02E0                 865 ;
02E0                 866 ;       NONE
02E0                 867 ;
02E0                 868 ; SIDE EFFECTS:
02E0                 869 ;
02E0                 870 ;       NONE
02E0                 871 ;
02E0                 872 ;--
```

```
                          0FFC  02E0   874 .ENTRY  DS_ERRSUP,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>   ;                    [14]
                                02E2   875
                     0FFF 8F    BB   02E2   876          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                          7E    DC   02E6   877          MOVPSL  -(SP)                       ; Save PSL for error typeout
                       10 AD    DD   02E8   878          PUSHL   16(FP)                      ; Push return address for typeout
                                02EB   879
          0000FE44'EF    04 AC    D0   02EB   880          MOVL    4(AP),L^DSA$GL_ERRNO        ; Error number to mailbox
                    50   0C AC    D0   02F3   881          MOVL    12(AP),R0                   ; Get address of ASCIC module name
                       51   80    9A   02F7   882          MOVZBL  (R0)+,R1                    ; Get length of string
             0000FE58'EF    51    D0   02FA   883          MOVL    R1,L^DSA$GL_DEVLEN          ; Move string length to mailbox
          52 0000FE5C'EF    9E   0301   884          MOVAB   L^DSA$GT_DEVNAM,R2
                                0308   885
                       82   80    90   0308   886 15$:     MOVB    (R0)+,(R2)+
                         FA 51    F5   030B   887          SOBGTR  R1,15$
                                C30E   888
                    52   08 BC    DE   030E   889          MOVAL   @8(AP),R2                   ; GET OPTIONAL MSG ADDRESS
                          07    12   0312   890          BNEQ    10$                         ; SKIP IF VALID
          52 00000074'EF    DE   0314   891          MOVAL   ANULL,R2                    ; ELSE FAKE A MESSAGE
                                031B   892
                                031B   893 10$:     $print  #ds$k_type_errsup, -        ; Print Supervisor error             [10]
                                031B   894                  #ds$k_printf, -            ; .. use PRINTF                      [10]
                                031B   895                  L^FMTERRSUP, -             ; .. Format                          [10]
                                031B   896                  12(AP), -                  ; .. force the message out           [08]
                                031B   897                  4(AP), -                   ; .. number                          [10]
                                031B   898                  #0, -                      ; .. current time                    [10]
                                031B   899                  R2                         ;                                    [08]
                                0338   900
     0000020A'EF    9F   0338   901          PUSHAB  L^FMTERRSUP2                ; Push address of register text      [08]
          7E    01    9A   033E   902          movzbl  #ds$k_printf, -(sp)         ; .. push PRINTF code                [10]
     FFFFFFF7 8F    DD   0341   903          pushl   #-ds$k_type_error_body      ; .. and type code (sticky)          [10]
00000000'EF    11    FB   0347   904          CALLS   #17,DSX$PRINT               ; Print the error                    [10]
                                034E   905
     00000000'EF    DF   034E   906          PUSHAL  L^DS$GT_VDS_Version         ; ... version                        [24]
     00000000'EF    DD   0354   907          PUSHL   L^DS$GL_Flags               ; ... DS flags                       [24]
     00000000'EF    DD   035A   908          PUSHL   L^DSA$GL_Flags              ; ... DSA flags                      [24]
     00000000'EF    DF   0360   909          PUSHAL  L^FmtErrSup1                ; ... format string                  [24]
          7E    01    9A   0366   910          MOVZBL  #DS$K_PrintF, -(SP)         ; ... use PRINTF                     [24]
          7E    04    98   0369   911          CVTBL   #DS$K_Type_ErrSup, -(SP)    ; Print Supervisor error             [24]
00000000'GF    06    FB   036C   912          CALLS   #6,G^DSX$PRINT              ;                                    [24]
                                0373   913
          52    5E    D0   0373   914          MOVL    SP,R2                       ; Copy current stack pointer
          53    08    D0   0376   915          MOVL    #8,R3                       ; Print several lines                [21]
                                0379   916
       52   20 A2    DE   0379   917 16$:     MOVAL   32(R2),R2                   ; Point past
          51    52    D0   037D   918          MOVL    R2,R1                       ; Copy
                                0380   919
          7E    71    7D   0380   920          MOVQ    -(R1),-(SP)                 ; Push quad
          7E    71    7D   0383   921          MOVQ    -(R1),-(SP)
          7E    71    7D   0386   922          MOVQ    -(R1),-(SP)
          7E    71    7D   0389   923          MOVQ    -(R1),-(SP)
                51    DD   038C   924          PUSHL   R1                          ; Address of stack
     0000029B'EF    9F   038E   925          PUSHAB  L^FMTERRSUP3                ; Edit text                          [08]
          7E    01    9A   0394   926          movzbl  #ds$k_printf,-(sp)          ; Use PRINTF                         [18]
          7E    09    98   0397   927          cvtbl   #ds$k_type_error_body,-(sp)     ;                                 [18]
00000000'GF    0C    FB   039A   928          CALLS   #12, G^DSX$PRINT            ; print it                           [18]
             D5 53    F5   03A1   929          SOBGTR  R3,16$                      ; Count and continue
                                03A4   930
```

```
                    04 AC    DD  03A4   931          pushl   4(ap)                          ; Set up for PRINT error number          [10]
          0000005C'EF        9F  03A7   932          pushab  L^ersuper                     ; .. 'Supervisor detected error'         [10]
          00000162'EF        9F  03AD   933          pushab  L^fmtenderror                 ; .. use end of error format             [10]
                    01       DD  03B3   934          pushl   #ds$k_printf                  ; .. use PRINTF                          [10]
                    0A       DD  03B5   935          pushl   #ds$k_type_error_end          ; .. print end of error                  [10]
     00000000'GF    05       FB  03B7   936          calls   #5, G^dsx$print               ; .. print it                            [10]
     0000FE00'EF    01       D3  03BE   937          BITL    #DSA$M_HALT, -                ;                                        [09]
                             03C5   938                   L^DSA$GL_FLAGS                 ; Halt on detection or isolation?
                    2A       13  03C5   939          BEQL    DSERRSUPX                     ; Branch if not
                             03C7   940
  00000000'EF   10 BD        DE  03C7   941          MOVAL   @16(FP),L^DS$AA_BPTADDR       ; SAVE RETURN 'PC'                       [08]
  00000000'EF   10 BD        90  03CF   942          MOVB    @16(FP),L^DS$AB_BPTINST       ; SAVE THE OPCODE LOCATED THERE          [08]
          10 BD    03        90  03D7   943          MOVB    #BPTCODE,@16(FP)              ; CAUSE A BREAK TO THE "CLI"
                             03DB   944          $PRINT  #ds$k_type_err_halt, -         ; Print error halt message               [10]
                             03DB   945                    #ds$k_printf, -              ; .. use PRINTF                          [10]
                             03DB   946                    L^FMTERRHLT, -               ; .. error halt format                   [10]
                             03DB   947                    16(FP)                       ; .. display halt PC                     [08]
                             03F1   948                                                 ;
                             03F1   949 DSERRSUPX:
     0000FE40'EF    05       D0  03F1   950          MOVL    #APM$_PRGERR,L^DSA$GL_MSGTYP    ; Indicate error type in mailbox
          00000000'EF        D6  03F8   951          Incl    L^Ds$GL_ErrSup_Count          ; Count this error                       [15]
                             03FE   952          Br_If_Not_APT   10$                   ; If not running from APT, branch        [19]
          00000000'EF    16  0406   953          JSB     L^APT_MSG                     ; Else, transfer control to APT          [08]
                             040C   954
                         04  040C   955 10$:     RET
```

```
                        040D    957          .SBTTL  DSR$Completion Routine
                        040D    958 ;++
                        040D    959 ; FUNCTIONAL DESCRIPTION:
                        040D    960 ;
                        040D    961 ;       This routine can be called unconditionally to print
                        040D    962 ;       the text for a given error code.  It returns immediately
                        040D    963 ;       if R0 does not indicate an error
                        040D    964 ;
                        040D    965 ; CALLING SEQUENCE:
                        040D    966 ;
                        040D    967 ;       BSBB    DSR$COMPLETION
                        040D    968 ;
                        040D    969 ; INPUT PARAMETERS:     NONE
                        040D    970 ;
                        C40D    971 ; IMPLICIT INPUTS:
                        040D    972 ;
                        040D    973 ;       R0      Contains the Completion code to be translated and typed
                        040D    974 ;
                        040D    975 ; OUTPUT PARAMETERS:    NONE
                        040D    976 ;
                        040D    977 ; IMPLICIT OUTPUTS:     NONE
                        040D    978 ;
                        040D    979 ; COMPLETION CODES:     NONE
                        040D    980 ;
                        040D    981 ; SIDE EFFECTS:         NONE... ALL REGISTERS ARE LEFT INTACT
                        040D    982 ;
                        040D    983 ;--
```

```
                                040D   985 DSR$COMPLETION::
                     01 50   E9 040D   986          BLBC    R0,10$                  ; Continue only if error
                             05 0410   987          RSB                             ; Return since not in error
                                0411   988 10$:
                          1F BB 0411   989          PUSHR   #^M<R0,R1,R2,R3,R4>     ; Save working registers
                       54 5E D0 0413   990          MOVL    SP,R4                   ; Copy current stack pointer
                                0416   991
                          50 DD 0416   992          PUSHL   R0                      ; Last arg is actual return
                       52 50 D0 0418   993          MOVL    R0,R2                   ; Save uncut code
                                041B   994
                       50 07 CA 041B   995          BICL    #7,R0                   ; Clear level bits
                 53 000002F0'EF 9E 041E   996          MOVAB   L^ERROR_CODES,R3        ; Point to table of error codes         [08]
                             61 10 0425   997          BSBB    80$                     ; Locate error text
               51 000003A4'EF41 3E 0427   998          MOVAW   L^ERROR_TEXT[R1],R1     ; Get offset to error text              [08]
                       7E 61 3C 042F   999          MOVZWL  (R1),-(SP)              ; Push offset to error text
                       6E 51 C0 0432  1000          ADDL    R1,(SP)                 ; And make it absolute
                                0435  1001
                51 52 02 01 EF 0435  1002          EXTZV   #1,#2,R2,R1             ; Get severity
                    00000894'EF41 9F 043A  1003          PUSHAB  L^SEVERITY[R1]          ; Push address of severity              [08]
                             01 DD 0441  1004          PUSHL   #1                      ; String length=1
                                0443  1005
                50 52 0C 10 EF 0443  1006          EXTZV   #16,#12,R2,R0           ; Get facility code
                 53 000002C5'EF 9E 0448  1007          MOVAB   L^UTILITY_CODES,R3      ; Point to table of utilities           [08]
                             2A 10 044F  1008          BSBB    50$                     ; Locate name of utility
               51 000002D5'EF41 3E 0451  1009          MOVAW   L^UTILITY_TEXT[R1],R1   ; Get offset to error text              [08]
                       7E 61 3C 0459  1010          MOVZWL  (R1),-(SP)              ; Push offset to error text
                       6E 51 C0 045C  1011          ADDL    R1,(SP)                 ; And make it absolute
                                045F  1012
                    000002AC'EF 9F 045F  1013          PUSHAB  L^T_ERROR_MASK          ; Push address for edit string          [08]
                       7E 01 9A 0465  1014          movzbl  #ds$k_printf, -(sp)     ; Use PRINTF                            [10]
                       7E 03 9A 0468  1015          movzbl  #ds$k_type_general_error, -(sp) ; And general error code        [10]
                       54 5E C2 046B  1016          SUBL    SP,R4                   ; Bytes used by args
                       54 04 C6 046E  1017          DIVL2   #4,R4                   ; Longwords
            00000000'GF 54 FB 0471  1018          CALLS   R4, G^DSX$PRINT         ; Type the error text                   [10]
                                0478  1019
                          1F BA 0478  1020          POPR    #^M<R0,R1,R2,R3,R4>     ; Restore
                             05 047A  1021          RSB                             ; Return
                                047B  1022 50$:
                       51 63 D0 047B  1023          MOVL    (R3),R1                 ; Get length of table
                    6341 50 D1 047E  1024 60$:     CMPL    R0, (R3)[R1]            ; Is this it?                           [12]
                             03 13 0482  1025          BEQL    70$                     ; Copy text address and return
                          F7 51 F5 0484  1026          SOBGTR  R1,60$                  ; Count and continue
                             05 0487  1027 70$:     RSB                             ; Return
                                0488  1028
                                0488  1029
                                0488  1030 80$:
                       51 63 D0 0488  1031          MOVL    (R3),R1                 ; Get length of table
                 7E    6341 07 CB 048B  1032 90$:     bicl3   #7, (r3)[r1], -(sp)     ; Clear out low bits                    [12]
                       8E 50 D1 0490  1033          CMPL    R0, (sp)+               ; Is this it?                           [12]
                             03 13 0493  1034          BEQL    100$                    ; Copy text address and return
                          F3 51 F5 0495  1035          SOBGTR  R1,90$                  ; Count and continue
                             05 0498  1036 100$:    RSB                             ; Return
                                0499  1037          .END
```

```
$$ARGS                          =  0000000A     D
$$N                             =  00000002     D
$$T1                            =  00000001     D
$$T2                            =  00000005     D
ANULL                              00000074 R   D   03
APC$_ABORT                      =  00000006     D
APC$_CONTINUE                   =  00000009     D
APC$_DUMP                       =  00000003     D
APC$_NOP                        =  00000000     D
APC$_START                      =  00000005     D
APC$_ZERO                       =  00000004     D
APM$_ABTDON                     =  00000006     D
APM$_DEVERR                     =  00000002     D
APM$_DONE                       =  0000000A     D
APM$_EXCEPT                     =  0000000B     D
APM$_HARDERR                    =  00000003     D
APM$_MORE                       =  00000008     D
APM$_NOMES                      =  00000000     D
APM$_PREPERR                    =  0000000C     D
APM$_PRGERR                     =  00000005     D
APM$_SOFTERR                    =  00000004     D
APM$_SPOOL                      =  00000009     D
APM$_SYSERR                     =  00000001     D
APT_MSG                            ********   X   04
BIT...                          =  00000004     D
BPTCODE                         =  00000003     D
CF$L_AP                            00000008     D
CF$L_FP                            0000000C     D
CF$L_ONCOND                        00000000     D
CF$L_PC                            00000010     D
CF$L_REG                           00000014     D
CF$W_MASK                          00000006     D
CF$W_PSW                           00000004     D
DEVFER                             00000019 R   D   03
DS$AA_BPTADDR                      ********   X   04
DS$AB_BPTINST                      ********   X   04
DS$GA_CHKLPPC                      ********   X   04
DS$GA_USER_ERROR_TEXT              ********   X   04
DS$GB_TYPECODE                     ********   X   04
DS$GL_DEVERR_COUNT                 ********   X   04
DS$GL_ERRCNT                       ********   X   04
DS$GL_ERRSUP_COUNT                 ********   X   04
DS$GL_FLAGS                        ********   X   04
DS$GL_HARDERR_COUNT                ********   X   04
DS$GL_PREPERR_COUNT                ********   X   04
DS$GL_SOFTERR_COUNT                ********   X   04
DS$GL_SYSERR_COUNT                 ********   X   04
DS$GPHARD                          ********   X   04
DS$GQ_TFSTID                       00000008 RG  D   02
DS$GT_VDS_VERSION                  ********   X   04
DS$GW_TTOUT                        ********   X   04
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_PRINTB                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  0000001B     D
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        =  0000000B     D
DS$K_TYPE_FIRST_PASS            =  00000011     D
DS$K_TYPE_GENERAL               =  00000000     D
DS$K_TYPE_GENERAL_ERROR         =  00000003     D
DS$K_TYPE_NO_TESTS              =  00000012     D
DS$K_TYPE_PARAM_ERROR           =  0000001C     D
DS$K_TYPE_PROGRAM_END           =  00000010     D
DS$K_TYPE_PROGRAM_INFO          =  00000017     D
DS$K_TYPE_PROGRAM_START         =  0000000F     D
DS$K_TYPE_QIO_INVADP            =  00000024     D
DS$K_TYPE_QIO_NODRIVER          =  00000022     D
DS$K_TYPE_QIO_WRONGVER          =  00000023     D
DS$K_TYPE_SCRIPT_ECHO           =  00000021     D
DS$K_TYPE_SCRIPT_PNF            =  0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         =  00000020     D
DS$K_TYPE_SCRIPT_SKIP           =  0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        =  00000019     D
DS$K_TYPE_START_ERR             =  00000018     D
DS$K_TYPE_START_LIST            =  00000025     D
DS$K_TYPE_SUMMARY               =  0000000E     D
DS$K_TYPE_USER_PROMPT           =  00000002     D
DS$K_WARNING                    =  00000000     D
DS$M_ABRTFLG                    =  00000040     D
DS$M_BADTIME                    =  00100000     D
DS$M_BATCH                      =  00400000     D
DS$M_BRKCLR                     =  00001000     D
DS$M_BRKPT                      =  00000800     D
DS$M_CHARFLG                    =  00000100     D
DS$M_CMDFLG                     =  00000080     D
DS$M_CTRLC                      =  00000001     D
DS$M_CTRLO                      =  00010000     D
DS$M_DEVFLG                     =  00000200     D
DS$M_DISABLCC                   =  01000000     D
DS$M_DONFLG                     =  00002000     D
DS$M_ERRFLG                     =  00000010     D
DS$M_EXCEPT                     =  00080000     D
DS$M_EXETST                     =  00040000     D
DS$M_HLTFLG                     =  00000008     D
```

DS$M_LODFLG	= 00000002	D	DS$_KRNLSTK	= 00660090	D		
DS$M_MEMMGT	= 00008000	D	DS$_LOGIC	= 00660070	D		
DS$M_OUTPUT	= 00800000	D	DS$_MCHK	= 00660088	D		
DS$M_RUBFLG	= 00000020	D	DS$_MMOFF	= 00660058	D		
DS$M_SCRIPT	= 00200000	D	DS$_NEEDUNIT	= 006600F8	D		
DS$M_SETIMR	= 02000000	D	DS$_NODE	= 00660128	D		
DS$M_STRFLG	= 00000004	D	DS$_NOPCS	= 00660110	D		
DS$M_SUBT	= 00004000	D	DS$_NORMAL	= 00660001	D		
DS$M_SYSFLG	= 00000400	D	DS$_NOSUPPORT	= 006600B1	D		
DS$M_TIMRON	= 00020000	D	DS$_NOTDON	= 00660030	D		
DS$V_ABRTFLG	= 00000006	D	DS$_NOTIMP	= 006600B0	D		
DS$V_BADTIME	= 00000014	D	DS$_NULLSTR	= 00660010	D		
DS$V_BATCH	= 00000016	D	DS$_OVERFLOW	= 00660008	D		
DS$V_BRKCLR	= 0000000C	D	DS$_POWER	= 00660098	D		
DS$V_BRKPT	= 0000000B	D	DS$_PROGERR	= 00660020	D		
DS$V_CHARFLG	= 00000008	D	DS$_SEVERE	= 00660004	D		
DS$V_CMDFLG	= 00000007	D	DS$_TRANSL	= 006600A0	D		
DS$V_CTRLC	= 00000000	D	DS$_TRUNCATE	= 00660028	D		
DS$V_CTRLO	= 00000010	D	DS$_UNEXPINT	= 006600D8	D		
DS$V_DEVFLG	= 00000009	D	DS$_VASFULL	= 00660048	D		
DS$V_DISABLCC	= 00000018	D	DS$_WARNING	= 00660000	D		
DS$V_DONFLG	= 0000000D	D	DSA$AL_APTMAIL	0000FE00	D		
DS$V_ERRFLG	= 00000004	D	DSA$AT_APTTXT	0000FA00	D		
DS$V_EXCEPT	= 00000013	D	DSA$GL_APTCOM	0000FE04	D		
DS$V_EXETST	= 00000012	D	DSA$GL_DEVLEN	0000FE58	D		
DS$V_HLTFLG	= 00000003	D	DSA$GL_ERRNO	0000FE44	D		
DS$V_LODFLG	= 00000001	D	DSA$GL_EVENT	0000FE48	D		
DS$V_MEMMGT	= 0000000F	D	DSA$GL_FLAGS	0000FE00	D		
DS$V_OUTPUT	= 00000017	D	DSA$GL_MSGTYP	0000FE40	D		
DS$V_RUBFLG	= 00000005	D	DSA$GL_PASSES	0000FE08	D		
DS$V_SCRIPT	= 00000015	D	DSA$GL_PASSNO	0000FE54	D		
DS$V_SETIMR	= 00000019	D	DSA$GL_SECTNO	0000FE10	D		
DS$V_STRFLG	= 00000002	D	DSA$GL_SID	0000FE14	D		
DS$V_SUBT	= 0000000E	D	DSA$GL_SUBTNO	0000FE4	D		
DS$V_SYSFLG	= 0000000A	D	DSA$GL_TESTNO	0000FE50	D		
DS$V_TIMRON	= 00000011	D	DSA$GL_UNITS	0000FE0C	D		
DS$_ARITH	= 006600D0	D	DSA$GQ_MSGPTR	0000FE68	D		
DS$_ASBE	= 00660118	D	DSA$GT_DEVNAM	0000FE5C	D		
DS$_BADLINK	= 006600F0	D	DSA$M_HALT	= 00000001	D		
DS$_BADTYPE	= 006600E8	D	DSA$V_APT	= 0000001F	D		
DS$_BIIC	= 00660120	D	DSA$V_BELL	= 00000003	D		
DS$_CHME	= 006600A8	D	DSA$V_HALT	= 00000000	D		
DS$_CHMK	= 006600E0	D	DSA$V_IE1	= 00000004	D		
DS$_DEVNAME	= 00660108	D	DSERRSUPX	000003F1	R	D	04
DS$_ERROR	= 00660002	D	DSQA$K_DISPAT_AFTER_INIT	= 00000014	D		
DS$_FHWE	= 00660068	D	DSQA$K_DISPAT_BEFORE_INIT	= 00000013	D		
DS$_FRAGBUF	= 00660080	D	DSQA$K_DISPAT_CALLTEST	= 00000015	D		
DS$_ICBUSY	= 006600C8	D	DSQA$K_DISPAT_DONE_TESTS	= 00000018	D		
DS$_ICERR	= 006600C0	D	DSQAK_DISPAT_DSXENDPASS_BGN	= 00000001	D		
DS$_IHWE	= 00660060	D	DSQAK_DISPAT_DSXENDPASS_END	= 00000002	D		
DS$_ILLCHAR	= 00660018	D	DSQAK_DISPAT_DSXENDPASS_MID	= 00000000	D		
DS$_ILLPAGCNT	= 00660078	D	DSQA$K_DISPAT_DS_CLEANUP_BGN	= 00000003	D		
DS$_ILLUNIT	= 00660100	D	DSQA$K_DISPAT_DS_CLEANUP_END	= 00000004	D		
DS$_INSFMEM	= 00660050	D	DSQA$K_DUMMY_T	= 00000007	D		
DS$_IPL2HI	= 006600B8	D	DSQA$K_ERROR_ERROR_BGN	= 00000006	D		
DS$_IVADDR	= 00660040	D	DSQA$K_ERROR_ERROR_END	= 00000005	D		
DS$_IVVECT	= 00660038	D	DSQA$K_KERNEL_INIT_CONTEXT	= 00000008	D		

```
DSQA$K_LOOP_DSX$BGNSUB          = 00000009     D
DSQA$K_LOOP_DSX$CKLOOP          = 0000000B     D
DSQA$K_LOOP_DSX$ENDSUB          = 0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS     = 0000000E     D
DSQA$K_PARAM_DSX$GETDATA        = 0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL     = 0000000F     D
DSQA$K_PARAM_DSX$GETSTRING      = 00000010     D
DSQA$K_PARAM_DSX$GETVIELD       = 0000000D     D
DSQA$K_QA_DSX$BRANCH            = 00000011     D
DSQA$K_START_BEFORE_HEADER      = 00000017     D
DSQA$K_START_VRRESTART          = 00000012     D
DSQA$K_START_VRSTART            = 00000016     D
DSR$CHECK_AUTOTEST_OFF_SET        ********   X   04
DSR$CHECK_MENUTEST_OFF_SET        ********   X   04
DSR$COMPLETION                    0000C40D RG D   04
DSX$ERRDEV                        00000075 RG D   04
DSX$ERRHARD                       00000027 RG D   04
DSX$ERRPREP                       0000004E RG D   04
DSX$ERRSOFT                       00000000 RG D   04
DSX$ERRSYS                        000000A3 RG D   04
DSX$PRINT                         ********   X   04
DS_ERRSUP                         000002E0 RG D   04
DS_TESTID                         00000274 RG D   04
ERR$_MSGADR                     = 0000000C     D
ERR$_NARGS                      = 0000000A     D
ERR$_NUM                        = 00000004     D
ERR$_P1                         = 00000014     D
ERR$_P2                         = 00000018     D
ERR$_P3                         = 0000001C     D
ERR$_P4                         = 00000020     D
ERR$_P5                         = 00000024     D
ERR$_P6                         = 00000028     D
ERR$_POINTER                    = 00000010     D
ERR$_UNIT                       = 00000008     D
ERROR_CODES                       000002F0 R   D   03
ERROR_TEXT                        000003A4 R   D   03
ERSUPER                           0000005C R   D   03
FMTENDERROR                       00000162 R   D   03
FMTERRHDR                         000000FF R   D   03
FMTERRHLT                         000000DD R   D   03
FMTERRSUP                         00000191 R   D   03
FMTERRSUP1                        000001D0 R   D   03
FMTERRSUP2                        0000020A R   D   03
FMTERRSUP3                        0000029B R   D   03
HARDER                            00000001 R   D   03
HP$A_DEPENDENT                    00000032     D
HP$A_DEVICE                       00000018     D
HP$A_DVA                          0000001C     D
HP$A_LINK                         00000020     D
HP$B_DRIVE                        00000008     D
HP$B_FLAGS                        0000000A     D
HP$Q_DEVICE                       00000000     D
HP$T_DEVICE                       0000000C     D
HP$T_TYPE                         00000026     D
HP$W_SIZE                         00000008     D
HP$W_VECTOR                       00000024     D
IO$_WRITEVBLK                     ********   X   04
L$A_CCP                           00000240     D
L$A_DEVP                          0000021C     D
L$A_DREG                          00000224     D
L$A_DTP                           00000218     D
L$A_ICP                           0000023C     D
L$A_LASTADR                       00000214     D
L$A_NAME                          00000208     D
L$A_REPP                          00000244     D
L$A_SECNAM                        00000250     D
L$A_STATAB                        00000248     D
L$A_TSTCNT                        00000254     D
L$L_ENVIRON                       00000204     D
L$L_ERRTYP                        0000024C     D
L$L_HEADLENGTH                    00000200     D
L$L_REV                           0000020C     D
L$L_UNIT                          00000220     D
L$L_UNUSED                        00000228     D
L$L_UPDATE                        00000210     D
MSGBELL                           00000000 R   D   03
PREPER                            0000002D R   D   03
QA$MAIN                           ********     X   04
QCLEANSEC                         000000A9 R   D   03
QFMTTESTID                        000000C0 R   D   03
QINITSEC                          0000008B R   D   03
QUNKDEV                           00000075 R   D   03
Q_BUFQWD                          00000000 R   D   02
RMS$_ACC                        = 0001C002     D
RMS$_CCR                        = 00018494     D
RMS$_DEV                        = 000184C4     D
RMS$_DME                        = 000184D4     D
RMS$_DNF                        = 0001C04A     D
RMS$_EOF                        = 0001827A     D
RMS$_FAB                        = 0001850C     D
RMS$_FNF                        = 00018292     D
RMS$_FNM                        = 0001852C     D
RMS$_IFI                        = 00018564     D
RMS$_IOP                        = 00018574     D
RMS$_IRC                        = 0001857C     D
RMS$_ISI                        = 00018584     D
RMS$_RAB                        = 0001863C     D
RMS$_RER                        = 0001C0F4     D
RMS$_RFA                        = 0001865C     D
RMS$_RTB                        = 000181A8     D
RRPTERROR                         000000D1 R   D   04
RRPTERRORX                        0000026A R   D   04
SEVERITY                          00000894 R   D   03
SIZ...                          = 00000001     D
SOFTER                            0000000D R   D   03
SS$_ACCVIO                        ********     X   03
SS$_CTRLERR                       ********     X   03
SS$_NOSUCHDEV                     ********     X   03
SS$_NOSUCHFILE                    ********     X   03
SYS$FAO                           ********     X   04
SYS$QIOW                          ********    GX   04
SYSFER                            00000048 R   D   03
TESTID_LEN                      = 00000020     D
T_ERROR_MASK                      000002AC R   D   03
```

```
T_TESTID                        00000010 R  D  02
UTILITY_CODES                   000002C5 R  D  03
UTILITY_TEXT                    000002D5 R  D  03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000030 (48.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000898 (2200.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000499 (1177.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=0000000A	159 (1)	159 (1)
$$N	=00000002	947 (1)	774 (1) 787 (1) 801 (1) 899 (1) 947 (1)
$$T1	=00000001	710 (1)	159 (1) 710 (1)
$$T2	=00000005	830 (1)	830 (1)
ANULL	00000074-R	201 (1)	737 (1) 891 (1)
APC$_ABORT	=00000006	153 (1)	
APC$_CONTINUE	=0C000C09	153 (1)	
APC$_DUMP	=00000003	153 (1)	
APC$_NOP	=00000000	153 (1)	
APC$_START	=00000005	153 (1)	
APC$_ZERO	=00000004	153 (1)	
APM$_ABTDON	=00000006	153 (1)	
APM$_DEVERR	=00000002	153 (1)	#-597 (1)
APM$_DONE	=0000000A	153 (1)	
APM$_EXCEPT	=0000000B	153 (1)	
APM$_HARDERR	=00000003	153 (1)	#-490 (1)
APM$_MORE	=00000008	153 (1)	
APM$_NOMES	=00000000	153 (1)	
APM$_PREPERR	=0000000C	153 (1)	#-542 (1)
APM$_PRGERR	=00000005	153 (1)	#-950 (1)
APM$_SOFTERR	=00000004	153 (1)	#-439 (1)
APM$_SPOOL	=00000009	153 (1)	
APM$_SYSERR	=00000001	153 (1)	#-652 (1)
APT_MSG	00000000-XR		442 (1) 493 (1) 545 (1) 600 (1) 655 (1) 953 (1)
BIT...	=00000004	163 (1)	154 (1) 155 (1) 157 (1) 163 (1)
BPTCODE	=00000003	150 (1)	#-797 (1) #-943 (1)
CFSL_PC	00000010		793 (1) #-795 (1) #-797 (1) #-801 (1)
DEVFER	00000019-R	190 (1)	592 (1)
DS$AA_BPTADDR	00000000-XR		#-794 (1) #-941 (1)
DS$AB_BPTINST	00000000-XR		#-796 (1) #-942 (1)
DS$GA_CHKLPPC	00000000-XR		#-704 (1)
DS$GA_USER_ERROR_TEXT	00000000-XR		#-757 (1)
DS$GB_TYPECODE	00000000-XR		#-779 (1)
DS$GL_DEVERR_COUNT	00000000-XR		#-591 (1)
DS$GL_ERRCNT	00000000-XR		#-702 (1)
DS$GL_ERRSUP_COUNT	00000000-XR		#-951 (1)
DS$GL_FLAGS	00000000-XR		594 (1) 649 (1) 703 (1) #-907 (1)
DS$GL_HARDERR_COUNT	00000000-XR		#-487 (1)
DS$GL_PREPERR_COUNT	00000000-XR		#-539 (1)
DS$GL_SOFTERR_COUNT	00000000-XR		#-436 (1)
DS$GL_SYSERR_COUNT	00000000-XR		#-646 (1)
DS$GPHARD	00000000-XR		712 (1)
DS$GQ_TESTID	00000008-R	174 (1)	#-774 (1) #-820 (1) #-824 (1) #-826 (1) #-827 (1) 830 (1)
DS$GT_VDS_VERSION	00000000-XR		906 (1)
DS$GW_TTOUT	00000000-XR		#-710 (1)
DS$K_ERROR	=00000002	157 (1)	
DS$K_NORMAL	=00000001	157 (1)	

```
DS$K_PRINTB                     =00000002       163     (1)
DS$K_PRINTF                     =00000001       163     (1)     #-1014  (1)     #-774   (1)     #-787   (1)     #-801   (1)
                                                                #-829   (1)     #-902   (1)     #-910   (1)     #-926   (1)
                                                                #-934   (1)     #-947   (1)
DS$K_PRINTI                     =00000000       163     (1)
DS$K_PRINTX                     =00000003       163     (1)
DS$K_SEVERE                     =00000004       157     (1)
DS$K_SUBSYS                     =00000066       157     (1)     157     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014       163     (1)
DS$K_TYPE_ABORT_TEST            =00000013       163     (1)
DS$K_TYPE_COMMAND_ERR           =00000015       163     (1)
DS$K_TYPE_COMMAND_OUT           =00000016       163     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A       163     (1)
DS$K_TYPE_DS_PROMPT             =00000001       163     (1)
DS$K_TYPE_DS_START              =00000C1D       163     (1)
DS$K_TYPE_ERRDEV                =00000008       163     (1)     191     (1)
DS$K_TYPE_ERRHARD               =00000006       163     (1)     185     (1)
DS$K_TYPE_ERROR_BODY            =00000009       163     (1)     #-778   (1)     #-903   (1)     #-927   (1)
DS$K_TYPE_ERROR_END             =0000000A       163     (1)     #-787   (1)     #-935   (1)
DS$K_TYPE_ERRPREP               =0000001B       163     (1)     194     (1)
DS$K_TYPE_ERRSOFT               =00000007       163     (1)     188     (1)
DS$K_TYPE_ERRSUP                =00000004       163     (1)     #-899   (1)     #-911   (1)
DS$K_TYPE_ERRSYS                =00000005       163     (1)     197     (1)
DS$K_TYPE_ERR_HALT              =0000000D       163     (1)     #-801   (1)     #-947   (1)
DS$K_TYPE_EXCEPTION             =0000000C       163     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B       163     (1)
DS$K_TYPE_FIRST_PASS            =00000011       163     (1)
DS$K_TYPE_GENERAL               =00000000       163     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003       163     (1)     #-1015  (1)
DS$K_TYPE_NO_TESTS              =00000012       163     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C       163     (1)
DS$K_TYPE_PROGRAM_END           =00000010       163     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017       163     (1)
DS$K_TYPE_PROGRAM_START         =0000000F       163     (1)
DS$K_TYPE_QIO_INVADP            =00000024       163     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022       163     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023       163     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021       163     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E       163     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020       163     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F       163     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019       163     (1)
DS$K_TYPE_START_ERR             =00000018       163     (1)
DS$K_TYPE_START_LIST            =00000025       163     (1)
DS$K_TYPE_SUMMARY               =0000000E       163     (1)
DS$K_TYPE_USER_PROMPT           =00000002       163     (1)
DS$K_WARNING                    =00000000       157     (1)
DS$M_ABRTFLG                    =00000040       154     (1)
DS$M_BADTIME                    =0010C000       154     (1)
DS$M_BATCH                      =00400000       154     (1)
DS$M_BRKCLR                     =00001000       154     (1)
DS$M_BRKPT                      =00000800       154     (1)
DS$M_CHARFLG                    =00000100       154     (1)
DS$M_CMDFLG                     =00000080       154     (1)
DS$M_CTRLC                      =00000001       154     (1)
DS$M_CTRLO                      =00010000       154     (1)
DS$M_DEVFLG                     =00000200       154     (1)
```

```
DS$M_DISABLCC                   =01000000       154     (1)
DS$M_DONFLG                     =00002000       154     (1)
DS$M_ERRFLG                     =00000010       154     (1)
DS$M_EXCEPT                     =00080000       154     (1)
DS$M_EXETST                     =00040000       154     (1)
DS$M_HLTFLG                     =00000008       154     (1)
DS$M_LODFLG                     =00000002       154     (1)
DS$M_MEMMGT                     =00008000       154     (1)
DS$M_OUTPUT                     =00800000       154     (1)
DS$M_RUBFLG                     =00000020       154     (1)
DS$M_SCRIPT                     =00200000       154     (1)
DS$M_SETIMR                     =02000000       154     (1)
DS$M_STRFLG                     =00000004       154     (1)
DS$M_SUBT                       =00004000       154     (1)
DS$M_SYSFLG                     =00000400       154     (1)
DS$M_TIMRON                     =00020000       154     (1)
DS$V_ABRTFLG                    =00000006       154     (1)
DS$V_BADTIME                    =00000014       154     (1)
DS$V_BATCH                      =00000016       154     (1)
DS$V_BRKCLR                     =0000000C       154     (1)
DS$V_BRKPT                      =0000000B       154     (1)
DS$V_CHARFLG                    =00000008       154     (1)
DS$V_CMDFLG                     =00000007       154     (1)
DS$V_CTRLC                      =00000000       154     (1)
DS$V_CTRLO                      =00000010       154     (1)
DS$V_DEVFLG                     =00000009       154     (1)     #-593     (1)
DS$V_DISABLCC                   =00000018       154     (1)
DS$V_DONFLG                     =0000000D       154     (1)
DS$V_ERRFLG                     =00000004       154     (1)     #-703     (1)
DS$V_EXCEPT                     =00000013       154     (1)
DS$V_EXETST                     =00000012       154     (1)
DS$V_HLTFLG                     =00000003       154     (1)
DS$V_LODFLG                     =00000001       154     (1)
DS$V_MEMMGT                     =0000000F       154     (1)
DS$V_OUTPUT                     =00000017       154     (1)
DS$V_RUBFLG                     =00000005       154     (1)
DS$V_SCRIPT                     =00000015       154     (1)
DS$V_SETIMR                     =00000019       154     (1)
DS$V_STRFLG                     =00000002       154     (1)
DS$V_SUBT                       =0000000E       154     (1)
DS$V_SYSFLG                     =0000000A       154     (1)     #-648     (1)
DS$V_TIMRON                     =00000011       154     (1)
DS$_ARITH                       =006600D0       157     (1)       295     (1)
DS$_ASBE                        =00660118       157     (1)
DS$_BADLINK                     =006600F0       157     (1)
DS$_BADTYPE                     =006600E8       157     (1)
DS$_BIIC                        =00660120       157     (1)
DS$_CHME                        =006600A8       157     (1)
DS$_CHMK                        =006600E0       157     (1)
DS$_DEVNAME                     =00660108       157     (1)
DS$_ERROR                       =00660002       157     (1)
DS$_FHWE                        =00660068       157     (1)       284     (1)
DS$_FRAGBUF                     =00660080       157     (1)       287     (1)
DS$_ICBUSY                      =006600C8       157     (1)
DS$_ICERR                       =006600C0       157     (1)       294     (1)
DS$_IHWE                        =00660060       157     (1)       283     (1)
DS$_ILLCHAR                     =00660018       157     (1)       276     (1)
```

```
DS$_ILLPAGCNT                   =00660078       157     (1)       286     (1)
DS$_ILLUNIT                     =00660100       157     (1)
DS$_INSFMEM                     =00660050       157     (1)       282     (1)
DS$_IPL2HI                      =006600B8       157     (1)       293     (1)
DS$_IVADDR                      =00660040       157     (1)       280     (1)
DS$_IVVECT                      =00660038       157     (1)       279     (1)
DS$_KRNLSTK                     =00660090       157     (1)       289     (1)
DS$_LOGIC                       =00660070       157     (1)       285     (1)
DS$_MCHK                        =00660088       157     (1)       288     (1)
DS$_MMOFF                       =00660058       157     (1)
DS$_NEEDUNIT                    =006600F8       157     (1)
DS$_NODE                        =00660128       157     (1)
DS$_NOPCS                       =00660110       157     (1)
DS$_NORMAL                      =00660001       157     (1)
DS$_NOSUPPORT                   =00660CB1       157     (1)
DS$_NOTDON                      =00660030       157     (1)
DS$_NOTIMP                      =006600B0       157     (1)       157     (1)       292     (1)
DS$_NULLSTR                     =00660010       157     (1)       275     (1)
DS$_OVERFLOW                    =00660008       157     (1)       274     (1)
DS$_POWER                       =00660098       157     (1)       290     (1)
DS$_PROGERR                     =00660020       157     (1)       277     (1)
DS$_SEVERE                      =00660004       157     (1)
DS$_TRANSL                      =006600A0       157     (1)       291     (1)
DS$_TRUNCATE                    =00660028       157     (1)       278     (1)
DS$_UNEXPINT                    =006600D8       157     (1)       296     (1)
DS$_VASFULL                     =00660048       157     (1)       281     (1)
DS$_WARNING                     =00660000       157     (1)       157     (1)
DSA$GL_DEVLEN                    0000FE58                         #-722     (1)     #-883     (1)
DSA$GL_ERRNO                     0000FE44                         #-706     (1)     #-774     (1)     #-787     (1)     #-880     (1)
DSA$GL_FLAGS                     0000FE00                           441     (1)       492     (1)       544     (1)       599     (1)
                                                                    654     (1)       707     (1)       732     (1)       782     (1)
                                                                    789     (1)       792     (1)     #-908     (1)     #-938     (1)
                                                                    952     (1)
DSA$GL_MSGTYP                    0000FE40                         #-440     (1)     #-491     (1)     #-543     (1)     #-598     (1)
                                                                  #-653     (1)     #-950     (1)
DSA$GL_PASSNO                    0000FE54                         #-774     (1)
DSA$GL_SUBTNO                    0000FE4C                         #-818     (1)     #-822     (1)     #-830     (1)
DSA$GL_TESTNO                    0000FE50                         #-815     (1)     #-830     (1)
DSA$GQ_MSGPTR                    0000FE68                         #-705     (1)
DSA$GT_DEVNAM                    0000FE5C                           726     (1)       884     (1)
DSA$M_HALT                      =00000001                        #-937     (1)
DSA$V_APT                       =0000001F                        #-441     (1)     #-492     (1)     #-544     (1)     #-599     (1)
                                                                  #-654     (1)     #-952     (1)
DSA$V_BELL                      =00000003                        #-707     (1)
DSA$V_HALT                      =00000000                        #-789     (1)     #-792     (1)
DSA$V_IE1                       =00000004                        #-732     (1)     #-782     (1)
DSERRSUPX                        000003F1-R     949     (1)      #-939     (1)
DSQA$K_DISPAT_AFTER_INIT        =00000014       155     (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013       155     (1)
DSQA$K_DISPAT_CALLTEST          =00000015       155     (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018       155     (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001       155     (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002       155     (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000       155     (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003       155     (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004       155     (1)
DSQA$K_DUMMY_T                  =00000007       155     (1)
```

```
DSQA$K_ERROR_ERROR_BGN          =00000006      155     (1)
DSQA$K_ERROR_ERROR_END          =00000005      155     (1)     #-804     (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008      155     (1)
DSQA$K_LOOP_DSX$BGNSUB          =00000009      155     (1)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B      155     (1)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A      155     (1)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E      155     (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C      155     (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F      155     (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010      155     (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D      155     (1)
DSQA$K_QA_DSX$BRANCH            =00000011      155     (1)
DSQA$K_START_BEFORE_HEADER      =00000017      155     (1)
DSQA$K_START_VRRESTART          =00000012      155     (1)
DSQA$K_START_VRSTART            =00000C16      155     (1)
DSR$CHECK_AUTOTEST_OFF_SET       00000000-XR                    751     (1)
DSR$CHECK_MENUTEST_OFF_SET       00000000-XR                    748     (1)
DSR$COMPLETION                   0000040D-R     985     (1)
DSX$ERRDEV                       00000075-R     589     (1)
DSX$ERRHARD                      00000027-R     485     (1)
DSX$ERRPREP                      0000004E-R     537     (1)
DSX$ERRSOFT                      00000000-R     434     (1)
DSX$ERRSYS                       000000A3-R     644     (1)
DSX$PRINT                        00000000-XR                    1018    (1)     774     (1)     787     (1)     801     (1)
                                                                899     (1)     904     (1)     912     (1)     928     (1)
                                                                936     (1)     947     (1)
DS_ERRSUP                        000002E0-R     874     (1)
DS_TESTID                        00000274-R     814     (1)     #-734     (1)
ERR$_MSGADR                     =0000000C       159     (1)       735     (1)
ERR$_NARGS                      =0000000A       159     (1)
ERR$_NUM                        =00000004       159     (1)     #-706     (1)
ERR$_P1                         =00000014       159     (1)
ERR$_P2                         =00000018       159     (1)
ERR$_P3                         =0000001C       159     (1)
ERR$_P4                         =00000020       159     (1)
ERR$_P5                         =00000024       159     (1)
ERR$_P6                         =00000028       159     (1)
ERR$_POINTER                    =00000010       159     (1)     #-776     (1)     780     (1)
ERR$_UNIT                       =00000008       159     (1)     #-712     (1)
ERROR_CODES                      000002F0-R     251     (1)       996     (1)
ERROR_TEXT                       000003A4-R     298     (1)       998     (1)
ERSUPER                          0000005C-R     199     (1)       932     (1)
FMTENDERROR                      00000162-R     218     (1)       787     (1)     933     (1)
FMTERRHDR                        000000FF-R     214     (1)       774     (1)
FMTERRHLT                        000000DD-R     212     (1)       801     (1)     947     (1)
FMTERRSUP                        00000191-R     220     (1)       899     (1)
FMTERRSUP1                       000001D0-R     223     (1)       909     (1)
FMTERRSUP2                       0000020A-R     226     (1)       901     (1)
FMTERRSUP3                       0000029B-R     231     (1)       925     (1)
HARDER                           00000001-R     184     (1)       488     (1)
HP$Q_DEVICE                      00000000                       #-715     (1)     #-716     (1)
IO$_WRITEVBLK                    00000000-XR                    #-710     (1)
L$A_NAME                         00000208                       #-774     (1)
L$L_REV                          0000020C                       #-774     (1)
L$L_UPDATE                       00000210                       #-774     (1)
MSGBELL                          00000000-R     192     (1)       710     (1)
PREPER                           0000002D-R     193     (1)       540     (1)
```

```
QA$MAIN                         00000000-XR                      804    (1)
QCLEANSEC                       000000A9-R    207    (1)     #-824    (1)
QFMTTESTID                      000000C0-R    209    (1)       830    (1)
QINITSEC                        0000008B-R    205    (1)     #-820    (1)
QUNKDEV                         00000075-R    203    (1)     #-720    (1)
Q_BUFQWD                        00000000-R    172    (1)     #-715    (1)     #-717    (1)     #-720    (1)     #-722    (1)
                                                             #-724    (1)     #-774    (1)
RMS$_ACC                       =0001C002                         260    (1)
RMS$_CCR                       =00018494                         261    (1)
RMS$_DEV                       =000184C4                         258    (1)
RMS$_DME                       =000184D4                         262    (1)
RMS$_DNF                       =0001C04A                         259    (1)
RMS$_EOF                       =0001827A                         263    (1)
RMS$_FAB                       =0001850C                         264    (1)
RMS$_FNF                       =00018292                         257    (1)
RMS$_FNM                       =0001852C                         273    (1)
RMS$_IFI                       =00018564                         265    (1)
RMS$_IUP                       =00018574                         266    (1)
RMS$_IRC                       =0001857C                         272    (1)
RMS$_ISI                       =00018584                         267    (1)
RMS$_RAB                       =0001863C                         268    (1)
RMS$_RER                       =0001C0F4                         269    (1)
RMS$_RFA                       =0001865C                         271    (1)
RMS$_RTB                       =000181A8                         270    (1)
RRPTERROR                       000000D1-R    701    (1)     #-438    (1)     #-489    (1)     #-541    (1)     #-596    (1)
                                                             #-651    (1)
RRPTERRORX                      0000026A-R    803    (1)     #-790    (1)
SEVERITY                        00000894-R    390    (1)      1003    (1)
SIZ...                         =00000001      154    (1)       154    (1)
SOFTER                          0000000D-R    187    (1)       437    (1)
SS$_ACCVIO                      00000000-XR                      253    (1)
SS$_CTRLERR                     00000000-XR                      256    (1)
SS$_NOSUCHDEV                   00000000-XR                      254    (1)
SS$_NOSUCHFILE                  00000000-XR                      255    (1)
SYS$FAO                         00000000-XR                      830    (1)
SYS$QIOW                        00000000-XR                      710    (1)
SYSFER                          00000048-R    196    (1)       647    (1)
TESTID_LEN                     =00000020      151    (1)       177    (1)     #-826    (1)
T_ERROR_MASK                    000002AC-R    234    (1)      1013    (1)
T_TESTID                        00000010-R    176    (1)       827    (1)
UTILITY_CODES                   000002C5-R    236    (1)      1007    (1)
UTILITY_TEXT                    000002D5-R    242    (1)      1009    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_PRINT_S	1	774 (1)	774 (1) 787 (1) 801 (1) 899 (1) 947 (1)
$DEF	1	163 (1)	
$DEFINI	1	156 (1)	156 (1) 158 (1) 160 (1) 161 (1) 162 (1)
$DS_CFDEF	1	156 (1)	156 (1)
$DS_DSADEF	5	158 (1)	158 (1)
$DS_DSDEF	2	157 (1)	157 (1)
$DS_ERRDEF	1	159 (1)	159 (1)
$DS_GPHARD_S	1	712 (1)	712 (1)
$DS_HDRDEF	2	160 (1)	160 (1)
$DS_HPODEF	2	161 (1)	161 (1)
$DS_TYPEDEF	4	163 (1)	163 (1)
$EQU	1	163 (1)	155 (1) 157 (1) 163 (1)
$EQULS1	1	163 (1)	155 (1) 157 (1) 163 (1)
$EQULST	1	155 (1)	155 (1) 157 (1) 163 (1)
$FAO_S	2	828 (1)	828 (1)
$GBLINI	2	154 (1)	154 (1) 155 (1) 157 (1) 163 (1)
$OFFDEF	1	159 (1)	159 (1)
$PRINT	2	762 (1)	762 (1) 783 (1) 798 (1) 893 (1) 944 (1)
$PUSHADR	1	710 (1)	710 (1) 830 (1)
$PUSHTWO	1	710 (1)	710 (1)
$QIOPUSH	1	710 (1)	710 (1)
$QIOW_S	1	708 (1)	708 (1)
$RMSDEF	11	162 (1)	162 (1)
$VIELD	1	154 (1)	154 (1)
$VIELD1	1	163 (1)	154 (1)
APTDEF	1	153 (1)	153 (1)
BR_IF_HALT	1	789 (1)	789 (1)
BR_IF_IE1	1	732 (1)	732 (1) 782 (1)
BR_IF_NOT_APT	1	441 (1)	441 (1) 492 (1) 544 (1) 599 (1) 654 (1) 952 (1)
BR_IF_NOT_BELL	1	707 (1)	707 (1)
DSFDEF	3	154 (1)	154 (1)
DSQA	2	155 (1)	155 (1)
QA_MAIN	1	804 (1)	804 (1)
SET_ERRFLG	1	703 (1)	703 (1)
SET_HALT	1	792 (1)	792 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.12	00:00:00.38
Command processing	144	00:00:00.72	00:00:01.62
Pass 1	825	00:00:14.20	00:00:20.86
Symbol table sort	3	00:00:00.89	00:00:01.06
Pass 2	261	00:00:03.53	00:00:04.63
Symbol table output	48	00:00:00.27	00:00:00.36
Psect synopsis output	7	00:00:00.03	00:00:00.03

Cross-reference output	94	00:00:01.39	00:00:03.66
Assembler run totals	1420	00:00:21.17	00:00:32.60

The working set limit was 1000 pages.
82479 bytes (162 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 602 non-local and 86 local symbols.
1037 source lines were read in Pass 1, producing 0 object records in Pass 2.
96 pages of virtual memory were used to define 37 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	9
DRB1:[DS.WORK]DS.MLB;218	11
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	32

756 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ERROR/UPDA=(ERROR.UPD,ERROR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

```
;    0001  0     %title '*** EVENT flag system services'
;    0002  0     module event (ident = '06-01',
;    0003  0                   addressing_mode (external = long_relative),
;    0004  0                   optlevel = 3,
;    0005  0                   optimize) =
;    0006  1     begin
;    0007  1
;    0008  1     !
;    0009  1     ! Copyright (c) 1977, 1981, 1983
;    0010  1     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0011  1     !
;    0012  1     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0013  1     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0014  1     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0015  1     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0016  1     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0017  1     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0018  1     ! REMAIN IN DEC.
;    0019  1     !
;    0020  1     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0021  1     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0022  1     ! CORPORATION.
;    0023  1     !
;    0024  1     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0025  1     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0026  1     !
;    0027  1     !++
;    0028  1     ! FACILITY:     VAX Diagnostic Supervisor
;    0029  1     !
;    0030  1     ! ABSTRACT:
;    0031  1     !
;    0032  1     ! ENVIRONMENT:
;    0033  1     !
;    0034  1     ! AUTHOR:       Dave Butenhof, 11-Sep-1981 [DS version 6.5]
;    0035  1     !
;    0036  1     ! MODIFIED BY:
;    0037  1     !
;    0038  1     !       00      - dave butenhof, 11-Sep-1981.  Converted module from Macro-32
;    0039  1     !                 to Bliss-32.  Primary purpose was to add SS$_WASSET and
;    0040  1     !                 SS$_WASCLR status returns to READEF, SETEF and CLREF.  Also,
;    0041  1     !                 routines were poorly structured and did not have entry masks.
;    0042  1     !
;    0043  1     !       01      Bob Bergazzi    May 17, 1983    Version 6.11
;    0044  1     !                 Changed the order of searching libraries.
;    0045  1     !--
```

```
;    0046  1     %sbttl 'Declarations'
;    0047  1
;    0048  1     !
;    0049  1     ! Include Files:
;    0050  1     !
;    0051  1     library '$diag';                !                                   [01]
;    0052  1
;    0053  1     library '$ds';                  ! DS-specific library               [01]
;    0054  1
;    0055  1     library 'sys$library:lib';      ! Common VMS library                [01]
;    0056  1
;    0057  1     !
;    0058  1     ! Macros:
;    0059  1     !
;    0060  1
;    0061  1     builtin
;    0062  1         testbitss,
;    0063  1         testbitsc;
;    0064  1
;    0065  1     !
;    0066  1     ! Equated Symbols:
;    0067  1     !
;    0068  1
;    0069  1     !
;    0070  1     ! Own Storage:
;    0071  1     !
;    0072  1
;    0073  1     !
;    0074  1     ! Externals:
;    0075  1     !
;    0076  1
;    0077  1     external
;    0078  1         ds$ax_softpcb,
;    0079  1         ds$gq_efl : bitvector [64]; ! The event flags
;    0080  1
;    0081  1     external routine
;    0082  1         kb_check : jsb_none;        ! Check for console input
;    0083  1
;    0084  1     !
;    0085  1     ! Psect definition
;    0086  1     !
;    0087  1
;    0088  1     psect code = code (share);
```

B 1 Dispatch Routine
C 1 Dispatch Routine
D 1 DSX$EndPass Routine
E 1 DSX$EndPass Routine
F 1 DSX$EndPass Routine
G 1 DS_Cleanup Routine
H 1 DS_Cleanup Routine
I 1 DS_Cleanup Routine
J 1 DSX$DoSummary and VRSummary Ro
K 1 DSX$DoSummary and VRSummary Ro
L 1 DSX$DoSummary and VRSummary Ro
M 1 VRShowStatus Routine
N 1 VRShowStatus Routine
B 2 Symbol table
C 2 Symbol table
D 2 Symbol table
E 2 Symbol table
F 2 Psect synopsis
G 2 Cross reference
H 2 Cross reference
I 2 Cross reference
J 2 Cross reference
K 2 Cross reference
L 2 Cross reference
M 2 Cross reference
N 2 Cross reference
B 3 Cross reference
C 3 Cross reference
D 3 - RL01/2 BOOT DRIVER
E 3 - RL01/2 BOOT DRIVER
F 3 DECLARATIONS
G 3 DECLARATIONS
H 3 RL01/2 Bootstrap driver code
I 3 RL01/2 Bootstrap driver code
J 3 RL01/2 Bootstrap driver code
K 3 RL01/2 Bootstrap driver code
L 3 Symbol table
M 3 Cross reference
N 3 Cross reference
B 4 - RK06/7 BOOT DRIVER
C 4 - RK06/7 BOOT DRIVER
D 4 DECLARATIONS
E 4 DECLARATIONS
F 4 DECLARATIONS
G 4 RK06/7 Bootstrap driver code
H 4 RK06/7 Bootstrap driver code
I 4 RK06/7 Bootstrap driver code
J 4 ECC - PERFORM ECC ERROR CORREC
K 4 ECC - PERFORM ECC ERROR CORREC
L 4 Symbol table
M 4 Cross reference
N 4 Cross reference
B 5 - RB730:RB02/RB80 BOOT DRIVER
C 5 - RB730:RB02/RB80 BOOT DRIVER
D 5 - RB730:RB02/RB80 BOOT DRIVER
E 5 DECLARATIONS
F 5 DECLARATIONS
G 5 DECLARATIONS
H 5 RB730:RB02/RB80 Bootstrap driv
I 5 RB730:RB02/RB80 Bootstrap driv
J 5 RB730:RB02/RB80 Bootstrap driv
K 5 RB730:RB02/RB80 Bootstrap driv
L 5 RB730:RB02/RB80 Bootstrap driv
M 5 Symbol table
N 5 Cross reference
B 6 Cross reference
C 6 Cross reference
D 6 - DR32 interrupt handler
E 6 - DR32 interrupt handler
F 6 DECLARATIONS
G 6 DECLARATIONS
H 6 DR$INT - DR32 INTERRUPT HANDLE
I 6 DR$INT - DR32 INTERRUPT HANDLE
J 6 Symbol table
K 6 Cross reference
L 6 Cross reference
M 6 *** DSLOAD DS$LOAD routine
N 6 *** DSLOAD DS$LOAD routine
B 7 *** DSLOAD DS$LOAD routine
C 7 DSX$LOAD routine
D 7 DSX$LOAD routine
E 7 DSX$LOAD routine
F 7 DSX$LOAD routine
G 7 DSX$LOAD routine
H 7 DSX$LOAD routine
I 7 DSX$LOAD routine
J 7 DSX$LOAD routine
K 7 *** DSVECS Module
L 7 *** DSVECS Module
M 7 *** DSVECS Module
N 7 *** DSVECS Module
B 8 Libraries and Macros
C 8 The Dispatch Vectors
D 8 The Dispatch Vectors
E 8 Symbol table
F 8 Cross reference
G 8 Cross reference
H 8 *** ENTRY DS service entry vec
I 8 *** ENTRY DS service entry vec
J 8 *** ENTRY DS service entry vec
K 8 *** ENTRY DS service entry vec
L 8 *** ENTRY DS service entry vec
M 8 *** ENTRY DS service entry vec
N 8 Libraries and Equated Symbols
B 9 Data Psect Declarations
C 9 DS Entry Points
D 9 DS Entry Points
E 9 DS Entry Points
F 9 DS Entry Points
G 9 DS Entry Points
H 9 DS Entry Points
I 9 DS Entry Points
J 9 DS Entry Points
K 9 DS Entry Points
L 9 DS Entry Points
M 9 DS Entry Points
N 9 DS Entry Points
B 10 DS Entry Points
C 10 DS Entry Points
D 10 DS Entry Points
E 10 DS Entry Points
F 10 DS Entry Points
G 10 DS Entry Points
H 10 DS Entry Points
I 10 DS Entry Points
J 10 DS Entry Points
K 10 DS Entry Points
L 10 DS Entry Points
M 10 DS Entry Points
N 10 DS Entry Points
B 11 DS Entry Points
C 11 DS Entry Points
D 11 DS Entry Points
E 11 DS Entry Points
F 11 DS Entry Points
G 11 DS Entry Points
H 11 DS Entry Points
I 11 DS Entry Points
J 11 DS Entry Points
K 11 DS Entry Points
L 11 DS Entry Points
M 11 DS Entry Points
N 11 DS Entry Points
B 12 DS$SRVDISP_TABLE - Dispatch ta
C 12 DSX$SERVICE_DISPATCHER - Servi
D 12 Resrvd_Entry Routine - For ent
E 12 Resrvd_Entry Routine - For ent
F 12 Symbol table
G 12 Symbol table
H 12 Symbol table
I 12 Psect synopsis
J 12 Cross reference
K 12 Cross reference
L 12 Cross reference
M 12 Cross reference
N 12 Cross reference
B 13 Cross reference
C 13 Cross reference
D 13 Cross reference
E 13 *** ERROR Error routines
F 13 *** ERROR Error routines
G 13 *** ERROR Error routines
H 13 *** ERROR Error routines
I 13 Libraries, Macros, Equated Sym
J 13 Work Psect Declarations
K 13 Data Psect Declarations
L 13 Data Psect Declarations
M 13 Data Psect Declarations
N 13 Data Psect Declarations
B 14 Data Psect Declarations
C 14 Data Psect Declarations
D 14 Data Psect Declarations
E 14 Data Psect Declarations
F 14 DSX$ErrSoft Routine
G 14 DSX$ErrSoft Routine
H 14 DSX$ErrHard Routine
I 14 DSX$ErrPrep Routine [17]
J 14 DSX$ErrPrep Routine [17]
K 14 DSX$ErrDev Routine
L 14 DSX$ErrDev Routine

M 14 DSX$ErrSys Routine
N 14 DSX$ErrSys Routine
B 15 RRtpError Routine
C 15 RRtpError Routine
D 15 RRtpError Routine
E 15 RRtpError Routine
F 15 DS_TestID Routine
G 15 DS_ErrSup Routine
H 15 DS_ErrSup Routine
I 15 DS_ErrSup Routine
J 15 DSR$Completion Routine
K 15 DSR$Completion Routine
L 15 Symbol table
M 15 Symbol table
N 15 Symbol table
B 16 Symbol table
C 16 Cross reference
D 16 Cross reference
E 16 Cross reference
F 16 Cross reference
G 16 Cross reference
H 16 Cross reference
I 16 Cross reference
J 16 Cross reference
K 16 *** EVENT flag system services
L 16 Declarations

```
;    0089  1     %sbttl 'check for legal event flag number'
;    0090  1
;    0091  1     routine check_efn : jsb_none =
;    0092  2         begin
;    0093  2
;    0094  2         builtin
;    0095  2             ap;
;    0096  2
;    0097  2         bind
;    0098  2             efn = .ap + 4;
;    0099  2
;    0100  2         if .efn gtr 127 then return ss$_illefc;
;    0101  2
;    0102  2         if .efn gtr 63 then return ss$_unasefc;
;    0103  2
;    0104  2         ss$_normal
;    0105  1         end;


                                                        .TITLE  EVENT *** EVENT flag system services
                                                        .IDENT  \06-01\

                                                        .EXTRN  DS$AX_SOFTPCB, DS$GQ_EFL
                                                        .EXTRN  KB_CHECK

                                                        .PSECT  CODE,NOWRT,  SHR,2

                 0000007F   8F       04    AC  D1 00000 CHECK_EFN:
                                                        CMPL    4(AP), #127                     ; 0100
                                           05  15 00008          BLEQ    1$                      ;
                            50       EC    8F  9A 0000A          MOVZBL  #236, R0                ;
                                           05  0000E            RSB                             ;
                            3F       04    AC  D1 0000F 1$:      CMPL    4(AP), #63              ; 0102
                                           06  15 00013          BLEQ    2$                      ;
                            50     0234    8F  3C 00016          MOVZWL  #564, R0                ;
                                           05  0001A            RSB                             ;
                            50             01  D0 0001B 2$:      MOVL    #1, R0                  ; 0105
                                           05  0001E            RSB                             ;

; Routine Size:  31 bytes,    Routine Base:  CODE + 0000
```

```
;    0106  1      %sbttl 'set event flag routine'
;    0107  1
;    0108  1      !++
;    0109  1      ! Functional Description:
;    0110  1      !
;    0111  1      !       Sets a single event flag specified by the caller
;    0112  1      !
;    0113  1      ! Calling Sequence:
;    0114  1      !
;    0115  1      !       status = $setef (efn)
;    0116  1      !
;    0117  1      ! Input Parameters:
;    0118  1      !
;    0119  1      !       efn =>  event flag number
;    0120  1      !
;    0121  1      ! Implicit Inputs:
;    0122  1      !
;    0123  1      !       None
;    0124  1      !
;    0125  1      ! Output Parameters:
;    0126  1      !
;    0127  1      !       None
;    0128  1      !
;    0129  1      ! Implicit Outputs:
;    0130  1      !
;    0131  1      !       None
;    0132  1      !
;    0133  1      ! Completion Codes:
;    0134  1      !
;    0135  1      !       SS$_WASCLR
;    0136  1      !       SS$_WASSET
;    0137  1      !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0138  1      !       SS$_ILLEFC                      ; EFN > 127
;    0139  1      !
;    0140  1      ! Side Effects:
;    0141  1      !
;    0142  1      !       None
;    0143  1      !
;    0144  1      !--
```

```
;    0145  1     global routine exe$setef (efn) =
;    0146  2         begin
;    0147  2
;    0148  2         (local s; if not (s = check_efn ()) then return .s);
;    0149  2
;    0150  2         testbitss (ds$gq_efl [.efn])^3 + 1
;    0151  1         end;


                                                 OFFC 00000          .ENTRY  EXE$SETEF, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0145
                                                                             R10,R11                                          ;
                                                 DD  10 00002        BSBB    CHECK_EFN                                        ; 0148
                                  12             50  E9 00004        BLBC    S, 2$                                            ;
                                                 50  D4 00007        CLRL    R0                                               ; 0150
                       02 00000000G  EF      04  AC  E3 00009        BBCS    EFN, DS$GQ_EFL, 1$                               ;
                                                 50  D6 00012        INCL    R0                                               ;
                                  50             08  C4 00014 1$:    MULL2   #8, R0                                           ;
                                                 50  D6 00017        INCL    R0                                               ;
                                                     04 00019 2$:    RET                                                      ; 0151

; Routine Size:  26 bytes,    Routine Base:  CODE + 001F
```

```
;    0152  1     %sbttl 'Clear event flag routine'
;    0153  1     !++
;    0154  1     ! Functional Description:
;    0155  1     !
;    0156  1     !       Clears a single event flag specified by the caller
;    0157  1     !
;    0158  1     ! Calling Sequence:
;    0159  1     !
;    0160  1     !       status = $clref (efn)
;    0161  1     !
;    0162  1     ! Input Parameters:
;    0163  1     !
;    0164  1     !       efn =>  event flag number
;    0165  1     !
;    0166  1     ! Implicit Inputs:
;    0167  1     !
;    0168  1     !       None
;    0169  1     !
;    0170  1     ! Output Parameters:
;    0171  1     !
;    0172  1     !       None
;    0173  1     !
;    0174  1     ! Implicit Outputs:
;    0175  1     !
;    0176  1     !       None
;    0177  1     !
;    0178  1     ! Completion Codes:
;    0179  1     !
;    0180  1     !       SS$_WASSET
;    0181  1     !       SS$_WASCLR
;    0182  1     !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0183  1     !       SS$_ILLEFC                      ; EFN > 127
;    0184  1     !
;    0185  1     ! Side Effects:
;    0186  1     !
;    0187  1     !       None
;    0188  1     !
;    0189  1     !--
;    0190  1     !
;    0191  1     global routine exe$clref (efn) =
;    0192  2         begin
;    0193  2         !
;    0194  2         (local s; if not (s = check_efn ()) then return .s);
;    0195  2         !
;    0196  2         testbitsc (ds$gq_efl [.efn])^3 + 1
;    0197  1         end;


                                        0FFC 00000          .ENTRY  EXE$CLREF, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0191
                                                                    R10,R11                                      ;
                                        C3  10 000C2        BSBB    CHECK_EFN                                    ; 0194
                                    12  50  E9 00004        BLBC    S, 2$                                        ;
                                        50  D4 00007        CLRL    R0                                           ; 0196
                 02 00000000G   EF  04  AC  E5 00009        BBCC    EFN, DS$GQ_EFL, 1$                           ;
```

```
                                          50  D6 00012          INCL    R0                                          ;
                                    50    08  C4 00014 1$:      MULL2   #8, R0                                      ;
                                          50  D6 00017          INCL    R0                                          ;
                                              04 00019 2$:      RET                                                 ; 0197

; Routine Size:  26 bytes,    Routine Base:  CODE + 0039
```

```
;    0198  1      %sbttl 'post event flag'
;    0199  1      !++
;    0200  1      ! Functional Description:
;    0201  1      !       Post event flag will set an event flag specified by number
;    0202  1      !       for a process specified by process ID (PID) and cause any
;    0203  1      !       necessary rescheduling if this event satisfies a wait
;    0204  1      !       condition for the process.
;    0205  1      !
;    0206  1      !       In the supervisor, this routine merely sets the flag and returns the
;    0207  1      !       PCB address as a side effect.
;    0208  1      !
;    0209  1      ! Calling Sequence:
;    0210  1      !
;    0211  1      !       JSB     SCH$POSTEF
;    0212  1      !
;    0213  1      ! Input Parameters:
;    0214  1      !       r1 - process identification (pid)
;    0215  1      !       r2 - priority increment class number
;    0216  1      !       r3 - event flag number
;    0217  1      !
;    0218  1      ! Implicit Inputs:
;    0219  1      !       None
;    0220  1      !
;    0221  1      ! Output Parameters:
;    0222  1      !       r0 - completion status code
;    0223  1      !       r4 - pcb address of process specified by pid
;    0224  1      !
;    0225  1      ! Implicit Outputs:
;    0226  1      !       event flag bit selected by pid and event flag number
;    0227  1      !       is set.
;    0228  1      !
;    0229  1      ! Completion Codes:
;    0230  1      !       SS$_NORMAL  -  normal successful completion
;    0231  1      !       SS$_NONEXPR -  non-existent process
;    0232  1      !
;    0233  1      ! Side Effects:
;    0234  1      !
;    0235  1      !       None
;    0236  1      !--
;    0237  1      !
;    0238  1      global routine sch$postef (pid, pri_inc, efn; pcb) : jsb_postef =
;    0239  2          begin
;    0240  2          pcb = ds$ax_softpcb;        ! Set PCB value for return
;    0241  2          ds$gq_efl [.efn] = 1;       ! Set flag
;    0242  2          ss$_normal
;    0243  1          end;


                                54 00000000G   EF   9E 00000 SCH$POSTEF::
                                                              MOVAB   DS$AX_SOFTPCB, PCB                      ; 0240
                     00 00000000G   EF                53   E2 00007                BBSS    EFN, DS$GQ_EFL, 1$      ; 0241
                                    50                01   D0 0000F 1$:           MOVL    #1, R0                  ; 0243
                                                           05 00012               RSB                             ;
```

; Routine Size: 19 bytes, Routine Base: CODE + 0053

```
;    0244  1     %sbttl 'read event flags routine'
;    0245  1
;    0246  1     !++
;    0247  1     ! Functional Description:
;    0248  1     !
;    0249  1     !       Returns a specified event flag cluster to a longword
;    0250  1     !       in the caller's area
;    0251  1     !
;    0252  1     ! Calling Sequence:
;    0253  1     !
;    0254  1     !       status =  ?adef (efn, state)
;    0255  1     !
;    0256  1     ! Input Paramete
;    0257  1     !
;    0258  1     !       efn =>    any event flag number within desired cluster
;    0259  1     !       state => longword address for returned cluster
;    0260  1     !
;    0261  1     ! Implicit Inputs:
;    0262  1     !
;    0263  1     !       None
;    0264  1     !
;    0265  1     ! Output Parameters:
;    0266  1     !
;    0267  1     !       state => longword pointed to receives all flags in cluseter
;    0268  1     !
;    0269  1     ! Implicit Outputs:
;    0270  1     !
;    0271  1     !       None
;    0272  1     !
;    0273  1     ! Completion Codes:
;    0274  1     !
;    0275  1     !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0276  1     !       SS$_ILLEFC                      ; EFN > 127
;    0277  1     !       SS$_WASSET                      ; Success, specified flag was set
;    0278  1     !       SS$_WASCLR                      ; Success, specified flag was clear
;    0279  1     !
;    0280  1     ! Side Effects:
;    0281  1     !
;    0282  1     !       None
;    0283  1     !
;    0284  1     !--
```

```
;    0285  1     global routine exe$readef (efn, state) =
;    0286  2         begin
;    0287  2
;    0288  2         (local s; if not (s = check_efn ()) then return .s);
;    0289  2
;    0290  3         (map ds$gq_efl : vector [2];        ! Access flags by longword
;    0291  3         .state = .ds$gq_efl [.efn/32]       ! Read correct longword
;    0292  2         );
;    0293  2
;    0294  2         .ds$gq_efl [.efn]^3 + 1     ! Return SS$_WASSET or SS_$WASCLR
;    0295  1         end;


                                                    0FFC 00000          .ENTRY  EXE$READEF, Save R2,R3,R4,R5,R6,R7,R8,R9,-      ; 0285
                                                                                R10,R11                                             ;
                                    52 00000000G    EF   9E 00002       MOVAB   DS$GQ_EFL, R2                                   ;
                                                    8F   10 00009       BSBB    CHECK_EFN                                       ; 0288
                                    15              50   E9 0000B       BLBC    S, 1$                                           ;
                        50      04  AC              20   C7 0000E       DIVL3   #32, EFN, R0                                    ; 0291
                                08  BC          6240   D0 00013       MOVL    DS$GQ_EFL[R0], @STATE                           ;
                50          62      01      04  AC   EF   00018       EXTZV   EFN, #1, DS$GQ_EFL, R0                          ; 0294
                                    50          08   C4 0001E       MULL2   #8, R0                                          ;
                                                50   D6 00021       INCL    R0                                              ;
                                                    04   00023 1$:    RET                                                     ; 0295

; Routine Size:  36 bytes,    Routine Base:  CODE + 0066
```

```
;    0296  1     %sbttl 'wait for single event flag routine'
;    0297  1
;    0298  1     !++
;    0299  1     ! Functional Description:
;    0300  1     !
;    0301  1     !       Suspends program execution until a single specified
;    0302  1     !       event flag sets
;    0303  1     !
;    0304  1     ! Calling Sequence:
;    0305  1     !
;    0306  1     !       state = $waitfr (efn)
;    0307  1     !
;    0308  1     ! Input Parameters:
;    0309  1     !
;    0310  1     !       efn => event flag number
;    0311  1     !
;    0312  1     ! Implicit Inputs:
;    0313  1     !
;    0314  1     !       None
;    0315  1     !
;    0316  1     ! Output Parameters:
;    0317  1     !
;    0318  1     !       None
;    0319  1     !
;    0320  1     ! Implicit Outputs:
;    0321  1     !
;    0322  1     !       None
;    0323  1     !
;    0324  1     ! Completion Codes:
;    0325  1     !
;    0326  1     !       SS$_NORMAL
;    0327  1     !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0328  1     !       SS$_ILLEFC                      ; EFN > 127
;    0329  1     !
;    0330  1     ! Side Effects:
;    0331  1     !
;    0332  1     !       None
;    0333  1     !
;    0334  1     !--
;    0335  1     !
;    0336  1     global routine exe$waitfr (efn) =
;    0337  2         begin
;    0338  2         !
;    0339  2         (local s; if not (s = check_efn ()) then return .s);
;    0340  2         !
;    0341  2         while not .ds$gq_efl [.efn] do kb_check ();
;    0342  2         ss$_normal
;    0343  1         end;


                                    0FFC 00000          .ENTRY  EXE$WAITFR, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0336
                                                                R10,R11                                         ;
                                FF71  30 00002          BSBW    CHECK_EFN                                   ; 0339
                          14      50  E9 00005          BLBC    S, 3$                                       ;
```

```
                    08 00000000G  EF        04    AC  E0 00008 1$:     BBS     EFN, DS$GQ_EFL, 2$                          ; 0341
                                     00000000G  EF  16 00011           JSB     KB_CHECK                                    ;
                                                EF  11 00017           BRB     1$                                          ;
                                  50            01  D0 00019 2$:     MOVL    #1, R0                                      ; 0343
                                                    04 0001C 3$:     RET                                                 ;

; Routine Size:  29 bytes,    Routine Base:  CODE + 0000
```

```
;    0344  1     %sbttl 'wait for logical or of event flags routine'
;    0345  1
;    0346  1     !++
;    0347  1     ! Functional Description:
;    0348  1     !
;    0349  1     !       Suspends program execution until any one of a set of
;    0350  1     !       specified event flags sets
;    0351  1     !
;    0352  1     ! Calling Sequence:
;    0353  1     !
;    0354  1     !       status = $wflor (efn, mask)
;    0355  1     !
;    0356  1     ! Input Parameters:
;    0357  1     !
;    0358  1     !       efn =>  any event flag number within desired cluster
;    0359  1     !       mask => mask of desired flags within the specified cluster
;    0360  1     !
;    0361  1     ! Implicit Inputs:
;    0362  1     !
;    0363  1     !       None
;    0364  1     !
;    0365  1     ! Output Parameters:
;    0366  1     !
;    0367  1     !       None
;    0368  1     !
;    0369  1     ! Implicit Outputs:
;    0370  1     !
;    0371  1     !       None
;    0372  1     !
;    0373  1     ! Completion Codes:
;    0374  1     !
;    0375  1     !       SS$_NORMAL
;    0376  1     !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0377  1     !       SS$_ILLEFC                      ; EFN > 127
;    0378  1     !
;    0379  1     ! Side Effects:
;    0380  1     !
;    0381  1     !       None
;    0382  1     !
;    0383  1     !--
;    0384  1
;    0385  1     global routine exe$wflor (efn, mask) =
;    0386  2         begin
;    0387  2
;    0388  2         while 1 do
;    0389  3             begin
;    0390  3
;    0391  3             local
;    0392  3                 cluster;
;    0393  3
;    0394  3             kb_check ();
;    0395  3
;    0396  3             (local s; if not (s = exe$readef (.efn, cluster)) then return .s);
;    0397  3
;    0398  3             if (.cluster and .mask) neq 0 then exitloop
;    0399  2             end;
;    0400  2
```

```
;   0401  2          ss$_normal
;   0402  1          end;


                                                 0FFC 00000          .ENTRY  EXE$WFLOR, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0385
                                                                             R10,R11                                          ;
                                5E                04  C2 00002           SUBL2   #4, SP                                   ;
                                   00000000G      EF  16 00005 1$:       JSB     KB_CHECK                                 ; 0394
                                                  5E  DD 00008           PUSHL   SP                                       ; 0396
                                         04       AC  DD 0000D           PUSHL   EFN                                      ;
                             AB  Ar               02  FB 00010           CALLS   #2, EXE$READEF                           ;
                                 09               50  E9 00014           BLBC    S, 2$                                    ;
                             08  AC               6E  D3 00017           BITL    CLUSTER, MASK                            ; 0398
                                                  E8  13 0001B           BEQL    1$                                       ;
                                 50               01  D0 0001D           MOVL    #1, R0                                   ; 0402
                                                  04  00020 2$:          RET                                              ;

; Routine Size:  33 bytes,    Routine Base:  CODE + 00A7
```

```
;    0403  1      %sbttl 'wait for logical and of event flags routine'
;    0404  1
;    0405  1      !++
;    0406  1      ! Functional Description:
;    0407  1      !
;    0408  1      !       Suspends program execution until all specified
;    0409  1      !       event flags are set
;    0410  1      !
;    0411  1      ! Calling Sequence:
;    0412  1      !
;    0413  1      !       status = $wfland (efn, mask)
;    0414  1      !
;    0415  1      ! Input Parameters:
;    0416  1      !
;    0417  1      !       efn =>  any event flag number within desired cluster
;    0418  1      !       mask => mask of desired flags within the specified cluster
;    0419  1      !
;    0420  1      ! Implicit Inputs:
;    0421  1      !
;    0422  1      !       None
;    0423  1      !
;    0424  1      ! Output Parameters:
;    0425  1      !
;    0426  1      !       None
;    0427  1      !
;    0428  1      ! Implicit Outputs:
;    0429  1      !
;    0430  1      !       None
;    0431  1      !
;    0432  1      ! Completion Codes:
;    0433  1      !
;    0434  1      !       SS$_NORMAL
;    0435  1      !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0436  1      !       SS$_ILLEFC                      ; EFN > 127
;    0437  1      !
;    0438  1      ! Side Effects:
;    0439  1      !
;    0440  1      !       None
;    0441  1      !
;    0442  1      !--
;    0443  1
;    0444  1      global routine exe$wfland (efn, mask) =
;    0445  2          begin
;    0446  2
;    0447  2          while 1 do
;    0448  3              begin
;    0449  3
;    0450  3              local
;    0451  3                  cluster;
;    0452  3
;    0453  3              kb_check ();
;    0454  3
;    0455  3              (local s; if not (s = exe$readef (.efn, cluster)) then return .s);
;    0456  3
;    0457  3              if (.cluster and .mask) eql .mask then exitloop
;    0458  3
;    0459  2              end;
```

```
;   0460  2
;   0461  2          ss$_normal
;   0462  1          end;
```

```
                                                    0FFC 00000          .ENTRY  EXE$WFLAND, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0444
                                                                                R10,R11                                     ;
                                              5E                    04  C2 00002          SUBL2   #4, SP                      ;
                                                 00000000G          EF  16 00005 1$:      JSB     KB_CHECK                    ; 0453
                                                                    5E  DD 0000B          PUSHL   SP                          ; 0455
                                                          04        AC  DD 0000D          PUSHL   EFN                         ;
                                           8A     AF                02  FB 00010          CALLS   #2, EXE$READEF              ;
                                                  11                50  E9 00014          BLBC    S, 2$                       ;
                                                  50        08      AC  D2 00017          MCOML   MASK, R0                    ; 0457
                                        50        6E                50  CB 0001B          BICL3   R0, CLUSTER, R0             ;
                                           08     AC                50  D1 0001F          CMPL    R0, MASK                    ;
                                                                    E0  12 00023          BNEQ    1$                          ;
                                                  50                01  D0 00025          MOVL    #1, R0                      ; 0462
                                                                        04 00028 2$:      RET                                 ;

; Routine Size:  41 bytes,    Routine Base:  CODE + 00C8
```

```
;   0463  1
;   0464  1      end
;   0465  0      eludom
```

```
;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  CODE                           241  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                                Library Statistics

;                                          --------  Symbols  --------      Pages       Processing
;       File                               Total     Loaded   Percent       Mapped      Time

;  DRB1:[DS.WORK]DIAG.L32;265                784          0         0           85        00:00.1
;  DRB1:[DS.WORK]DS.L32;159                  653          2         0           42        00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7           18017          3         0          975        00:04.6
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE EVENT

; Size:         241 code + 0 data bytes
; Run Time:        00:10.4
; Elapsed Time:    00:15.8
; Lines/CPU Min:    2687
; Lexemes/CPU-Min:  4190
; Memory Used:  55 pages
; Compilation Complete
```

Table of contents

(1) 131 Libraries, External & Equated Symbols, Macros
(1) 177 Work Psect Declarations
(1) 186 Data Psect Declarations
(1) 316 COND_HANDLR ROUTINE - HANDLE UNWANTED EXCEPTIONS
(1) 429 DSX$PrintSig Routine - Format Signal Array
(1) 812 FIND_USERPC Find USER PC on stack
(1) 865 TST$MChk Routine - Machine Check Handler ; [14]
(1) 921 HARDWARE EXCEPTION ENTRY POINTS
(1) 1255 SEARCH FOR CONDITION HANDLER
(1) 1381 DSX$SETPRIEXV - ESTABLISH PRIMARY EXCEPTION VECTOR [19]

```
0000     1          .TITLE  EXCEPT  *** EXCEPT Machine Exception handling
0000     2          .IDENT  /07-24/
0000     3          .NoShow Conditionals                ;                                   [18]
0000     4          .DSABL  GBL
0000     5
0000     6  ;++
0000     7  ; Copyright (c) 1977, 1982, 1983, 1984
0000     8  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     9  ;
0000    10  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    11  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    12  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    13  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    14  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    15  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    16  ; REMAIN IN DEC.
0000    17  ;
0000    18  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    19  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    20  ; CORPORATION.
0000    21  ;
0000    22  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    23  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    24  ;--
```

```
                        0000     26 ;++
                        0000     27 ; FACILITY:
                        0000     28 ;
                        0000     29 ; ABSTRACT:
                        0000     30 ;
                        0000     31 ; ENVIRONMENT:
                        0000     32 ;
                        0000     33 ; AUTHOR:        ROGER RIGGS     15-AUG-78       VERSION 01.
                        0000     34 ;
                        0000     35 ; MODIFICATIONS:
                        0000     36 ;                Roger Riggs     07-FEB-79       VERSION 01. (EVSAA 5.00)
                        0000     37 ;       01       Added conversion of T-bit trap in compatability mode
                        0000     38 ;                to compatability mode trap with code=7.
                        0000     39 ;       02       Fixed lower limit of user stack in SEARCH routine
                        0000     40 ;       03       Removed ISTK/KSTK Switching stuff.  Interrupts occur on
                        0000     41 ;                the correct stack as indicated by the SCB
                        0000     42 ;       04       Fixed up position of carriage prior to calling CLI.
                        0000     43 ;                Make sure its on a new line at the left margin
                        0000     44 ;
                        0000     45 ;                Roger Riggs     11-Sept-1979
                        0000     46 ;       05       Added TST$MCHK Machine check entry point for
                        0000     47 ;                testing the existence of a memory location
                        0000     48 ;
                        0000     49 ;                Roger Riggs     29-OCT-1979
                        0000     50 ;       06       More cleanup and added DS$PRINTSIG, which can
                        0000     51 ;                be used by the program to print the signal array of an
                        0000     52 ;                exception
                        0000     53 ;
                        0000     54 ;                Roger Riggs, 14-Feb-1980
                        0000     55 ;       07       Added QIO to cancel Control O before printing exception info
                        0000     56 ;
                        0000     57 ;                Roger Riggs, 21-Feb-1980
                        0000     58 ;       08       Added duplicate defs for entry points to satisify internal debugger
                        0000     59 ;                Also added code to revert to old handler for arithmetic traps
                        0000     60 ;                and faults.
                        0000     61 ;
                        0000     62 ;                Roger Riggs, 16-Apr-1980
                        0000     63 ;       09       Corrected bug that reports the incorrect value in R3 during
                        0000     64 ;                exception call to CLI
                        0000     65 ;
                        0000     66 ;                Roger Riggs, 21-Jun-1980, Version 5.5
                        0000     67 ;       10       Changed DSX$PRINTSIG to use PRINTB for First line
                        0000     68 ;                and PRINTX for info lines.  Added new entry point
                        0000     69 ;                DSX$PRINTSIGI that uses PRINTI for all output.
                        0000     70 ;
                        0000     71 ;                Dave Butenhof, 11-sep-1980, Version 6.0
                        0000     72 ;       11       Fixed link-time truncation errors in module (change
                        0000     73 ;                word-relative to long-relative)
                        0000     74 ;
                        0000     75 ;                Roger Riggs, 17-Sep-1980
                        0000     76 ;       12       Added Type and VA to error typeout for translation not
                        0000     77 ;
                        0000     78 ;                Dave Butenhof, 23-apr-1981, version 6.4
                        0000     79 ;       13       Correct spelling in some messages, change to mixed case.
                        0000     80 ;
                        0000     81 ;       14       - Jack Stansbury, 22-Oct-1981, Version 6.-
                        0000     82 ;                Changed SEP Psect to CODE. Fixed truncation errors.
```

```
                                0000     83 ;                Also added .LIBRARY statements for $DS and $DIAG.
                                0000     84 ;
                                0000     85 ;        15      - Dave Butenhof, 16-Nov-1981, Version 6.5
                                0000     86 ;                Split the FIND_USERPC routine into two--one will actually
                                0000     87 ;                find the last user PC and return the address (FIND_USER_PC),
                                0000     88 ;                while the other (FIND_USERPC) will call the first, then
                                0000     89 ;                format and type the response.
                                0000     90 ;
                                0000     91 ;        16      - Dave Butenhof, 17-Dec-1981, Version 6.6
                                0000     92 ;                Correct a bug in TST$MCHK which seems to have been here all
                                0000     93 ;                along--it clears off machine check logout incorrectly.
                                0000     94 ;
                                0000     95 ;        17      - Dave Butenhof, 09-Mar-1982, version 6.6
                                0000     96 ;                Fix PRINTSIG to know about SS$_MCHECK code...new for V3a.
                                0000     97 ;
                                0000     98 ;        18      - Jack Stansbury, 23-Mar-1982, Version 6.8
                                0000     99 ;                Added typecoding to all the messages. Also made a few minor
                                0000    100 ;                enhancements (e.g., taking out the .LIST statements, changing
                                0000    101 ;                the text in the .SBTTL's, etc.).
                                0000    102 ;
                                0000    103 ;        19      - Richard Brown, 27-May-82, Version 6.8
                                0000    104 ;                Addition of routine DSX$SETPRIEXV, which is used to establish
                                0000    105 ;                a primary exception vector.
                                0000    106 ;
                                0000    107 ;        20      Jack Stansbury, 8-Mar-1983, Version 6.11
                                0000    108 ;                Took out the Find_User_PC routine and put it into the new
                                0000    109 ;                CallFrame.B32 module.
                                0000    110 ;
                                0000    111 ;        21      Bob Bergazzi    5-May-1983      Version 6.11
                                0000    112 ;                Added array single bit error interrupt (vector 064) reporting for
                                0000    113 ;                XXX processor.
                                0000    114 ;
                                0000    115 ;        22      Bob Bergazzi    Feb. 3, 1984    Version 6.14
                                0000    116 ;                Changed FMT_EXCEPT and FMT_UNEXPINT to say ..."SCB vector"
                                0000    117 ;                instead of ..."vector". This makes it clearer to some people.
                                0000    118 ;
                                0000    119 ;        23      Domenic Andella 8-Mar-1984      Version 6.14
                                0000    120 ;                Remove ASBE handler from DSX$PRINTSIG routine to
                                0000    121 ;                MCHKXXX module, routine ASBE_WORK.
                                0000    122 ;
                                0000    123 ;        24      Domenic Andella 1-May-1984      Version 7.0
                                0000    124 ;                Include FAULTCLEAR selection macro, and push/pop
                                0000    125 ;                of selection code to each call to DSR$FAULT_CLEAR in
                                0000    126 ;                routines EXE_MCHK, and TST$MCHK. This allows cpu
                                0000    127 ;                and routine dependent action to occur.
                                0000    128 ;
                                0000    129 ;--
```

```
                        0000    131          .SBTTL  Libraries, External & Equated Symbols, Macros
                        0000    132 ;
                        0000    133 ;       INCLUDE FILES:
                        0000    134 ;
                        0000    135
                        0000    136          .Library        /Sys$Library:Lib/       ;                                       [15]
                        0000    137          .Library        /$DS/                   ;                                       [14]
                        0000    138          .Library        /$Diag/                 ;                                       [14]
                        0000    139
                        0000    140 ;
                        0000    141 ; EXTERNAL SYMBOLS
                        0000    142 ;
                        0000    143
                        0000    144          .EXTRN  DSX$PRINTI,     DS_ERRSUP,      DS$ABORT,       BPTMAX
                        0000    145          .EXTRN  DS$AA_BPTADDR,  DS$GA_BREAKVEC, DS$GA_TBITVEC,  DEBUG$GB_FLAGS
                        0000    146          .EXTRN  CHR$MCHK,       DS$CLI,         COND_DEBUG,     SYS$UNWIND
                        0000    147          .EXTRN  MODELST,        APSLMNE,        DS$CVTREG,      USTKPTR
                        0000    148          .EXTRN  PCB_BASE,       DS$GL_FLAGS,    STACK_BASE,     DSX$PRINTB
                        0000    149          .EXTRN  SYS$CALL_HANDL, DSR$FAULT_CLEAR,SCRIPT$STOP,    DSX$PRINTX
                        0000    150          .EXTRN  ISTKPTR,        ESTKPTR,        SSTKPTR,        IO$M_CANCTRLO
                        0000    151          .EXTRN  IO$_WRITEVBLK,  DS$GW_TTOUT,    DS$GB_TypeCode                  ;[18
                        0000    152          .EXTRN  SCB_IMAGE,      DSR$Find_User_PC                                ;[20
                        0000    153
                        0000    154 ;
                        0000    155 ;       EQUATED SYMBOLS:
                        0000    156 ;
                        0000    157
                        0000    158          $DS_TypeDef                     ; Define typecodes                      [18]
                        0000    159          $DS_CVTREG_DEF
                        0000    160          $DS_DSDEF
                        0000    161          $PRDEF
                        0000    162          $PSLDEF
                        0000    163          $DS_SCBDEF
                        0000    164          $SSDEF
                        0000    165          CLIDEF
                        0000    166          CMKDEF
                        0000    167          DSFDEF
                        0000    168          APTDEF
                        0000    169          $DS_DSADEF
                        0000    170          $CHFDEF
                        0000    171          $SSDEF
                        0000    172          FLTCLR_SEL
00000001                0000    173          IS      = 1
00000100                0000    174          BUFSIZ  = 256                                   ;
                        0000    175
```

```
                        0000   177          .SBTTL  Work Psect Declarations
                    00000000   178          .PSECT  Work, NoExe, NoShr, Wrt, Long                ;                [14]
                        0000   179
00000004                0000   180 NEW_SP:          .BLKL   1
                        0004   181
0000000C                0004   182 NEW_AP_FP:       .BLKQ   1
                        000C   183
00000014                000C   184 NEW_PC_PSL:      .BLKQ   1
```

```
                                0014   186          .SBTTL  Data Psect Declarations
                            00000000   187          .PSECT  Data, NoExe, Shr, NoWrt, Long                    ;                [14]
                                0000   188
                                0000   189          MODNAM  EXCEPT
                                0007   190
                                0007   191 CTL$AQ_EXCVEC:   ; PRIMARY,SECONDARY EXCEPTION VECTORS FOR EXCEPTION HANDLERS
                  00000000'00000000'   0007   192          .ADDRESS        0,COND_DEBUG    ; KERNEL EXCEPTION HANDLER
                  00000000'00000000'   000F   193          .ADDRESS        0,COND_DEBUG    ; EXEC   EXCEPTION HANDLER
                  00000000'00000000'   0017   194          .ADDRESS        0,COND_DEBUG    ; SUPER  EXCEPTION HANDLER
                  00000000'00000000'   001F   195          .ADDRESS        0,COND_DEBUG    ; USER   EXCEPTION HANDLER
                                0027   196
                                0027   197 CTL$AL_STACK:    ; ADDRESS OF TOP OF EACH MODE STACK INDEX BY LONGWORD MODE
00000000'00000000'00000000'00000000'   0027   198          .ADDRESS        ISTKPTR,ESTKPTR,SSTKPTR,USTKPTR
```

```
                        0037   200 ;+
                        0037   201 ; These are the messages output by the exception printing routines. These do    [18]
                        0037   202 ; not need to contain typecodes since the typecode will be set explicitly by    [18]
                        0037   203 ; the routines, and the appropriate print routine will be called (rather than   [18]
                        0037   204 ; calling Print).                                                               [18]
                        0037   205 ;-
                        0037   206
                        0037   207 Msg_UMchk:      $PrintI_L       Fmt_UMchk                       ;              [17]
                        003F   208 MSG_KRNLSTK:    $PRINTI_L       FMT_EXCEPT, T_KRNLSTK, T_ABORT, <<^X08>>
                        0053   209 MSG_POWER:      $PRINTI_L       FMT_EXCEPT, T_POWER, T_INTRPT, <<^X0C>>
                        0067   210 MSG_OPCDEC:     $PRINTI_L       FMT_EXCEPT, T_OPCDEC, T_FAULT, <<^X10>>
                        007B   211 MSG_OPCCUS:     $PRINTI_L       FMT_EXCEPT, T_OPCCUS, T_FAULT, <<^X14>>
                        008F   212 MSG_ROPRAND:    $PRINTI_L       FMT_EXCEPT, T_ROPRAND, T_FLT_ABO, <<^X18>>
                        00A3   213 MSG_RADRMOD:    $PRINTI_L       FMT_EXCEPT, T_RADRMOD, T_FAULT, <<^X1C>>
                        00B7   214 MSG_ACCVIO:     $PRINTI_L       FMT_EXCEPT, T_ACCVIO, T_FAULT, <<^X20>>
                        00CB   215 MSG_TRANSL:     $PRINTI_L       FMT_EXCEPT, T_TRANSL, T_FAULT, <<^X24>>
                        00DF   216 MSG_TBIT:       $PRINTI_L       FMT_EXCEPT, T_TBIT, T_TRAP, <<^X28>>
                        00F3   217 MSG_BREAK:      $PRINTI_L       FMT_EXCEPT, T_BREAK, T_FAULT, <<^X2C>>
                        0107   218 MSG_COMPAT:     $PRINTI_L       FMT_EXCEPT, T_COMPAT, T_FLT_ABO, <<^X30>>
                        011B   219 MSG_CHMK:       $PRINTI_L       FMT_EXCEPT, T_CHMK, T_TRAP, <<^X40>>
                        012F   220 MSG_CHME:       $PRINTI_L       FMT_EXCEPT, T_CHME, T_TRAP, <<^X44>>
                        0143   221 MSG_CHMS:       $PRINTI_L       FMT_EXCEPT, T_CHMS, T_TRAP, <<^X48>>
                        0157   222 MSG_CHMU:       $PRINTI_L       FMT_EXCEPT, T_CHMU, T_TRAP, <<^X4C>>
                        016B   223
```

```
                                00000188' 016B    225 AAT_ARITH::     .ADDRESS        0$
                                00000198' 016F    226                 .ADDRESS        1$
                                000001AC' 0173    227                 .ADDRESS        2$
                                000001C3' 0177    228                 .ADDRESS        3$
                                000001D5' 017B    229                 .ADDRESS        4$
                                000001ED' 017F    230                 .ADDRESS        5$
                                00000200' 0183    231                 .ADDRESS        6$
                                00000211' 0187    232                 .ADDRESS        7$
                                          018B    233
6E 65 70 20 50 41 52 54 20 6F 4E 00' 018B    234 0$:     .ASCIC  .No TRAP pending.           ;  [13]
                   67 6E 69 64  0197
                                0F  018B
65 76 6F 20 72 65 67 65 74 6E 49 00' 019B    235 1$:     .ASCIC  .Integer overflow.          ;  [13]
                      77 6F 6C 66 72  01A7
                                10  C19B
76 69 64 20 72 65 67 65 74 6E 49 00' 01AC    236 2$:     .ASCIC  .Integer divide by zero.    ;  [13]
   6F 72 65 7A 20 79 62 20 65 64 69  01B8
                                16  01AC
76 6F 20 67 6E 69 74 61 6F 6C 46 00' 01C3    237 3$:     .ASCIC  .Floating overflow.         ;  [13]
                         77 6F 6C 66 72 65  01CF
                                11  01C3
69 64 20 67 6E 69 74 61 6F 6C 46 00' 01D5    238 4$:     .ASCIC  .Floating divide by zero.   ;  [13]
6F 72 65 7A 20 79 62 20 65 64 69 76  01E1
                                17  01D5
6E 75 20 67 6E 69 74 61 6F 6C 46 00' 01ED    239 5$:     .ASCIC  .Floating underflow.        ;  [13]
                      77 6F 6C 66 72 65 64  01F9
                                12  01ED
65 76 6F 20 6C 61 6D 69 63 65 44 00' 0200    240 6$:     .ASCIC  .Decimal overflow.          ;  [13]
                         77 6F 6C 66 72  020C
                                10  0200
76 69 64 20 6C 61 6D 69 63 65 44 00' 0211    241 7$:     .ASCIC  .Decimal divide by zero.    ;  [13]
   6F 72 65 7A 20 79 62 20 65 64 69  021D
                                16  0211
```

```
                0228   243 MSG_INTOVF:     $PRINTI_L       FMT_EXCEPT, T_INTOVF, T_TRAP, <<^X34>>
                023C   244 MSG_INTDIV:     $PRINTI_L       FMT_EXCEPT, T_INTDIV, T_TRAP, <<^X34>>
                0250   245 MSG_FLTOVF:     $PRINTI_L       FMT_EXCEPT, T_FLTOVF, T_TRAP, <<^X34>>
                0264   246 MSG_FLTDIV:     $PRINTI_L       FMT_EXCEPT, T_FLTDIV, T_TRAP, <<^X34>>
                0278   247 MSG_FLTUND:     $PRINTI_L       FMT_EXCEPT, T_FLTUND, T_TRAP, <<^X34>>
                028C   248 MSG_DECOVF:     $PRINTI_L       FMT_EXCEPT, T_DECOVF, T_TRAP, <<^X34>>
                02A0   249 MSG_SUBRNG:     $PRINTI_L       FMT_EXCEPT, T_SUBRNG, T_TRAP, <<^X34>>
                02B4   250 MSG_FLTOVF_F:   $PRINTI_L       FMT_EXCEPT, T_FLTOVF, T_FAULT, <<^X34>>
                02C8   251 MSG_FLTDIV_F:   $PRINTI_L       FMT_EXCEPT, T_FLTDIV, T_FAULT, <<^X34>>
                02DC   252 MSG_FLTUND_F:   $PRINTI_L       FMT_EXCEPT, T_FLTUND, T_FAULT, <<^X34>>
```

```
                                00000310' 02F0   254 AAT_COMPAT:     .ADDRESS        0$              ; RESERVED OPCODE FAULT
                                00000326' 02F4   255                 .ADDRESS        1$              ; BPT FAULT
                                00000330' 02F8   256                 .ADDRESS        2$              ; IOT FAULT
                                0000033A' 02FC   257                 .ADDRESS        3$              ; EMT FAULT
                                00000344' 0300   258                 .ADDRESS        4$              ; TRAP FAULT
                                0000034F' 0304   259                 .ADDRESS        5$              ; ILLEGAL INSTRUCTION FAULT
                                00000369' 0308   260                 .ADDRESS        6$              ; ODD ADDRESS ABORT
                                0000037B' 030C   261                 .ADDRESS        7$              ; UNUSED
70 6F 20 64 65 76 72 65 73 65 52 00' 0310   262 0$:                 .ASCIC  .Reserved opcode fault. ;                               [13]
      74 6C 75 61 66 20 65 64 6F 63  031C
                                  15  0310
      74 6C 75 61 66 20 54 50 42 00' 0326   263 1$:                 .ASCIC  .BPT fault.             ;                               [13]
                                  09  0326
      74 6C 75 61 66 20 54 4F 49 00' 0330   264 2$:                 .ASCIC  .IOT fault.             ;                               [13]
                                  09  0330
      74 6C 75 61 66 20 54 4D 45 00' 033A   265 3$:                 .ASCIC  .EMT fault.             ;                               [13]
                                  09  033A
   74 6C 75 61 66 20 50 41 52 54 00' 0344   266 4$:                 .ASCIC  .TRAP fault.            ;                               [13]
                                  0A  0344
73 6E 69 20 6C 61 67 65 6C 6C 49 00' 034F   267 5$:                 .ASCIC  .Illegal instruction fault.     ;                       [13]
75 61 66 20 6E 6F 69 74 63 75 72 74  035B
                                  74 6C  0367
                                  19  034F
73 73 65 72 64 64 61 20 64 64 4F 00' 0369   268 6$:                 .ASCIC  .Odd address abort.     ;                               [13]
                     74 72 6F 62 61 20  0375
                                  11  0369
            6E 77 6F 6E 6B 6E 55 00' 037B   269 7$:                 .ASCIC  .Unknown.               ;                               [13]
                                  07  037B
```

```
61 6D 20 53 4D 56 20 3F 3F 2F 21 00' 0383   271 Fmt_UMchk:       .AsciC  '!/?? VMS machine check (hardware' -    ;                [17]
20 6B 63 65 68 63 20 65 6E 69 68 63  038F
6F 6C 20 65 72 61 77 64 72 61 68 28  039B
61 76 61 20 74 6F 6E 20 74 75 6F 67  03A7
       2F 21 3A 29 65 6C 62 61 6C 69  03B3
                                 39  0383
                                     03BD   272                             ' logout not available):!/'              ;        [17]
41 21 20 43 41 21 20 3F 3F 2F 21 00' 03BD   273 FMT_EXCEPT:      .ASCIC  '!/?? !AC !AC through SCB vector: !XB(X)!/'    ;        [22]
43 53 20 68 67 75 6F 72 68 74 20 43  03C9
58 21 20 3A 72 6F 74 63 65 76 20 42  03D5
              2F 21 29 58 28 42 29   03E1
                                     03BD
65 70 78 65 6E 55 20 3F 3F 2F 21 00' 03E7   274 FMT_UNEXPINT:    .ASCIC  '!/?? Unexpected trap or interrupt thru SCB vector !XW!/' ;[
72 6F 20 70 61 72 74 20 64 65 74 63  03F3
74 20 74 70 75 72 72 65 74 6E 69 20  03FF
74 63 65 76 20 42 43 53 20 75 72 68  040B
           2F 21 57 58 21 20 72 6F  0417
                                 37  03E7
72 6F 72 72 65 20 74 61 20 43 50 00' 041F   275 FMT_ERRPC:       .ASCIC  'PC at error:!_!_!XL(X)!/'               ;        [13]
21 29 58 28 4C 58 21 5F 21 5F 21 3A  042B
                                 2F  0437
                                 18  041F
6E 72 75 74 65 72 20 72 65 73 55 00' 0438   276 FMT_USRPC:       .ASCIC  'User return PC:!_!_!XL(X)!/'            ;        [13]
28 4C 58 21 5F 21 5F 21 3A 43 50 20  0444
                       2F 21 29 58  0450
                                 18  0438
6E 72 75 74 65 72 20 72 65 73 55 00' 0454   277 Fmt_User_PC_NF: .Ascic  'User return PC: none found!!!/'          ;        [20]
6F 66 20 65 6E 6F 6E 20 3A 43 50 20  0460
              2F 21 21 21 64 6E 75   046C
                                 1E  0454
6F 72 72 65 20 74 61 20 4C 53 50 00' 0473   278 FMT_ERRPSL:      .ASCIC  'PSL at error:!_!_!XL(X)!_; !AC!/'       ;        [13]
29 58 28 4C 58 21 5F 21 5F 21 3A 72  047F
           2F 21 43 41 21 20 3B 5F 21  048B
                                 20  0473
64 64 61 20 6C 61 75 74 72 69 56 00' 0494   279 FMT_VIRT:        .ASCIC  'Virtual address:!_!XL(X)!/Fault type:!_!_!XL(X)!/' ; [13]
58 28 4C 58 21 5F 21 3A 73 73 65 72  04A0
70 79 74 20 74 6C 75 61 46 2F 21 29  04AC
29 58 28 4C 58 21 5F 21 5F 21 3A 65  04B8
                       2F 21 31 29  04C4
                                 31  0494
63 20 43 41 21 20 43 41 21 5F 21 00' 04C6   280 FMT_EXCEP_CODE: .ASCIC  '!_!AC !AC code:!_!AC!/'                 ;        [13]
   2F 21 43 41 21 5F 21 3A 65 64 6F  04D2
                                 16  04C6
21 5F 21 5F 21 5F 21 3A 67 72 41 00' 04DD   281 FMT_CHMX_ARG:    .ASCIC  'Arg:!_!_!_!XL(X)!/'                     ;        [13]
              2F 21 29 58 28 4C 58   04E9
                                 12  04DD
20 6E 77 6F 6E 6B 6E 55 20 3F 3F 00' 04F0   282 FMT_UNK_ARG:     .ASCIC  '?? Unknown exception/condition:!/'      ;        [13]
6F 63 2F 6E 6F 69 74 70 65 63 78 65  04FC
       2F 21 3A 6E 6F 69 74 69 64 6E  0508
                                 21  04F0
21 5F 21 5F 21 3A 74 6E 75 6F 43 00' 0512   283 FMT_UNK_CNT:     .ASCIC  'Count:!_!_!_!XL(X)!/'                   ;        [13]
           2F 21 29 58 28 4C 58 21 5F  051E
                                 14  0512
5F 21 5F 21 5F 21 3A 4C 55 21 50 00' 0527   284 FMT_PXXX:        .ASCIC  'P!UL:!_!_!_!XL(X)!/'
              2F 21 29 58 28 4C 58 21  0533
                                 13  0527
```

```
63 61 74 73 20 4C 45 4E 52 45 4B 00' 053B   286 T_KRNLSTK:      .ASCIC  'KERNEL stack not valid'       ;                    [13]
   64 69 6C 61 76 20 74 6F 6E 20 6B  0547
                                 16  053B
   6C 69 61 66 20 72 65 77 6F 50 00' 0552   287 T_POWER:       .ASCIC  'Power fail'            ;                    [13]
                                 0A  0552
72 70 2F 64 65 76 72 65 73 65 52 00' 055D   288 T_OPCDEC:      .ASCIC  'Reserved/privileged instruction'  ;         [13]
73 6E 69 20 64 65 67 65 6C 69 76 69  0569
               6E 6F 69 74 63 75 72 74  0575
                                 1F  055D
65 72 20 72 65 6D 6F 74 73 75 43 00' 057D   289 T_OPCCUS:      .ASCIC  'Customer reserved instruction' ;            [13]
72 74 73 6E 69 20 64 65 76 72 65 73  0589
                  6E 6F 69 74 63 75  0595
                                 1D  057D
70 6F 20 64 65 76 72 65 73 65 52 00' 059B   290 T_ROPRAND:     .ASCIC  'Reserved operand'      ;                    [13]
                     64 6E 61 72 65  05A7
                                 10  059B
64 61 20 64 65 76 72 65 73 65 52 00' 05AC   291 T_RADRMOD:     .ASCIC  'Reserved addressing mode'      ;            [13]
64 6F 6D 20 67 6E 69 73 73 65 72 64  05B8
                                 65  05C4
                                 18  05AC
74 6E 6F 63 20 73 73 65 63 63 41 00' 05C5   292 T_ACCVIO:      .ASCIC  'Access control violation'      ;            [13]
6F 69 74 61 6C 6F 69 76 20 6C 6F 72  05D1
                              6E 6F  05DD
                                 18  05C5
6E 6F 69 74 61 6C 73 6E 61 72 54 00' 05DE   293 T_TRANSL:      .ASCIC  'Translation not valid' ;                    [13]
      64 69 6C 61 76 20 74 6F 6E 20  05EA
                                 15  05DE
                  65 63 61 72 54 00' 05F4   294 T_TBIT:        .ASCIC  'Trace'                 ;                    [13]
                                 05  05F4
   74 6E 69 6F 70 6B 61 65 72 42 00' 05FA   295 T_BREAK:       .ASCIC  'Breakpoint'            ;                    [13]
                                 0A  05FA
65 64 6F 6D 20 65 67 6E 61 68 43 00' 0605   296 T_CHMK:        .ASCIC  'Change mode KERNEL'    ;                    [13]
               4C 45 4E 52 45 4B 20  0611
                                 12  0605
65 64 6F 6D 20 65 67 6E 61 68 43 00' 0618   297 T_CHME:        .ASCIC  'Change mode EXECUTIVE' ;                    [13]
      45 56 49 54 55 43 45 58 45 20  0624
                                 15  0618
65 64 6F 6D 20 65 67 6E 61 68 43 00' 062E   298 T_CHMS:        .ASCIC  'Change mode SUPERVISOR'        ;            [13]
   52 4F 53 49 56 52 45 50 55 53 20  063A
                                 16  062E
65 64 6F 6D 20 65 67 6E 61 68 43 00' 0645   299 T_CHMU:        .ASCIC  'Change mode USER'      ;                    [13]
                  52 45 53 55 20  0651
                                 10  0645
20 64 65 74 63 65 70 78 65 6E 55 00' 0656   300 T_XCHFAILURE:  .ASCIC  'Unexpected status from condition handler'      ;  [13]
20 6D 6F 72 66 20 73 75 74 61 74 73  0662
61 68 20 6E 6F 69 74 69 64 6E 6F 63  066E
               72 65 6C 64 6E 00 00  067A
                                 28  0656
                  70 61 72 54 00' 067F   301 T_TRAP:        .ASCIC  'Trap'                  ;                    [13]
                                 04  067F
                  74 6C 75 61 46 00' 0684   302 T_FAULT:       .ASCIC  'Fault'                 ;                    [13]
                                 05  0684
                  74 72 6F 62 41 00' 068A   303 T_ABORT:       .ASCIC  'Abort'                 ;                    [13]
                                 05  068A
      74 70 75 72 72 65 74 6E 49 00' 0690   304 T_INTRPT:      .ASCIC  'Interrupt'             ;                    [13]
                                 09  0690
74 72 6F 62 61 2F 74 6C 75 61 46 00' 069A   305 T_FLT_ABO:     .ASCIC  'Fault/abort'           ;                    [13]
```

```
                                                  0B  069A
69 6C 69 62 61 74 61 70 6D 6F 43 00'  06A6   306 T_COMPAT:    .ASCIC  'Compatability mode'    ;                          [13]
                  65 64 6F 6D 20 79 74  06B2
                                        12  06A6
74 73 20 73 75 6F 69 76 65 72 50 00'  06B9   307 T_STKVIO:    .ASCIC  'Previous stack access violation'    ;             [13]
76 20 73 73 65 63 63 61 20 6B 63 61   06C5
               6E 6F 69 74 61 6C 6F 69  06D1
                                        1F  06B9
65 76 6F 20 72 65 67 65 74 6E 49 00'  06D9   308 T_INTOVF:    .ASCIC  'Integer overflow'      ;                          [13]
                        77 6F 6C 66 72  06E5
                                        10  06D9
76 69 64 20 72 65 67 65 74 6E 49 00'  06EA   309 T_INTDIV:    .ASCIC  'Integer divide by zero'    ;                      [13]
   6F 72 65 7A 20 79 62 20 65 64 69   06F6
                                        16  06EA
76 6F 20 67 6E 69 74 61 6F 6C 46 00'  0701   310 T_FLTOVF:    .ASCIC  'Floating overflow'     ;                          [13]
                     77 6F 6C 66 72 65  070D
                                        11  0701
65 64 2F 67 6E 69 74 61 6F 6C 46 00'  0713   311 T_FLTDIV:    .ASCIC  'Floating/decimal divide by zero'    ;             [13]
65 64 69 76 69 64 20 6C 61 6D 69 63   071F
               6F 72 65 7A 20 79 62 20  072B
                                        1F  0713
6E 75 20 67 6E 69 74 61 6F 6C 46 00'  0733   312 T_FLTUND:    .ASCIC  'Floating underflow'    ;                          [13]
                     77 6F 6C 66 72 65 64  073F
                                        12  0733
65 76 6F 20 6C 61 6D 69 63 65 44 00'  0746   313 T_DECOVF:    .ASCIC  'Decimal overflow'      ;                          [13]
                        77 6F 6C 66 72  0752
                                        10  0746
72 20 74 70 69 72 63 73 62 75 53 00'  0757   314 T_SUBRNG:    .ASCIC  'Subscript range'       ;                          [13]
                           65 67 6E 61  0763
                                        0F  0757
```

```
                     0767   316          .SBTTL  COND_HANDLR ROUTINE - HANDLE UNWANTED EXCEPTIONS
                00000000    317          .PSECT  CODE, EXE, SHR, NOWRT, LONG
                     0000   318 ;++
                     0000   319 ; FUNCTIONAL DESCRIPTION:
                     0000   320 ;
                     0000   321 ;       This routine handles all exceptions not wanted by other handlers.
                     0000   322 ;       It could be considered a last chance exception handler and is setup
                     0000   323 ;       as such in user mode.
                     0000   324 ;
                     0000   325 ; CALLING SEQUENCE:
                     0000   326 ;
                     0000   327 ;       PROCEEDURE CALL
                     0000   328 ;
                     0000   329 ; INPUT PARAMETERS:
                     0000   330 ;
                     0000   331 ;       STANDARD MECHANISM AND SIGNAL ARRAY ARGS
                     0000   332 ;
                     0000   333 ; IMPLICIT INPUTS:
                     0000   334 ;
                     0000   335 ;       NONE
                     0000   336 ;
                     0000   337 ; OUTPUT PARAMETERS:
                     0000   338 ;
                     0000   339 ;       NONE
                     0000   340 ;
                     0000   341 ; IMPLICIT OUTPUTS:
                     0000   342 ;
                     0000   343 ;       NONE
                     0000   344 ;
                     0000   345 ; COMPLETION CODES:
                     0000   346 ;
                     0000   347 ;       SS$_CONTINUE -or-
                     0000   348 ;       SS$_RESIGNAL
                     0000   349 ;
                     0000   350 ; SIDE EFFECTS:
                     0000   351 ;
                     0000   352 ;       NONE
                     0000   353 ;--
```

```
                          000C  0000   355 .ENTRY  COND_HANDLR,^M<R2,R3>           ; Entry point to handle exceptions
                                0002   356
                                0002   357         $QIO_S  FUNC=#IO$_WRITEVBLK!IO$M_CANCTRLO, -
                                0002   358                 CHAN=DS$GW_TTOUT        ; Cancel Control-O
          0000011A'EF    04 BC   FA   0023   359         CALLG   @4(AP),DSX$PRINTSIGI    ; Format signal array
                         01 50   E8   002B   360         BLBS    R0,COND_HANDLR_CLI      ; Branch if known condition
                                 04   002E   361         RET                             ; Return with resignal status
                                      002F   362
                                      002F   363 COND_HANDLR_CLI:
                                      002F   364         Br_If_User      5$              ; Branch if in user mode                [18]
          0000FE40'EF    0B   D0   0037   365         MOVL    #APM$_EXCEPT, -         ; Tell APT about the error
                                      003E   366                 DSA$GL_MSGTYP
                                      003E   367
          00000000'EF    0C   90   003E   368 5$:     MovB    #DS$K_Type_Exception, - ; Set typecode                          [18]
                                      0045   369                 L^DS$GB_TypeCode        ;                                       [18]
              03CD'CF    6C   FA   0045   370         CALLG   (AP),W^FIND_USERPC      ; Locate and print PC<10000
               00000000'EF    D4   004A   371         ClrL    L^DS$GB_TypeCode        ; Clear typecode                        [18]
                                      0050   372
                                      0050   373 ;+
                                      0050   374 ; INFORMATION ABOUT THE UNEXPECTED CONDITION HAS BEEN PRINTED.
                                      0050   375 ; CALL CLI TO ALLOW THE OPERATOR TO EXAMINE THE SITUATION.
                                      0050   376 ; DETERMINE THE STACK POINTER AT THE TIME OF THE INTERRUPT
                                      0050   377 ;-
                                      0050   378
                50    04 BC   DE   0050   379         MOVAL   @4(AP),R0               ; ADDRESS OF SIGNAL ARRAY
                   51    60   D0   0054   380         MOVL    (R0),R1                 ; GET COUNT OF ARGS
                50    FC A041   DE   0057   381         MOVAL   -4(R0)[R1],R0           ; POINT AT PC/PSL PAIR
                  53    08 A0   DE   005C   382         MOVAL   8(R0),R3                ; Stack pointer before exception
                22 04 A0    1A   E0   0060   383         BBS     #PSL$V_IS,4(R0), 10$    ; IF IS WAS SET THAT WAS CORRECT
                                      0065   384         Br_If_User      10$             ; Branch if in user mode                [18]
                                      006D   385
        51    04 A0    02   18   EF   006D   386         EXTZV   #PSL$V_CURMOD, #PSL$S_CURMOD, -
                                      0073   387                 4(R0),R1                ; GET PREVIOUS MODE
                            52   DC   0073   388         MOVPSL  R2                      ; Get current PSL
                 52    02   18   ED   0075   389         CMPZV   #PSL$V_CURMOD,#PSL$S_CURMOD, -
                            51         0079   390                 R2,R1                   ; Same mode as previous?
                            0B   13   007A   391         BEQL    10$                     ; Use stack pointer in R3 if so
           53    00000000'EF41   D0   007C   392         MOVL    PCB_BASE[R1],R3         ; Get SP from PCB
                      53    51   DB   0084   393         MFPR    R1,R3                   ; Get implementation dependent SP
                                      0087   394
                      7E    60   7D   0087   395 10$:     MOVQ    (R0),-(SP)              ; STACK PC/PSL
                            08   BB   008A   396         PUSHR   #^M<R3>                 ; Push exception SP
                   7E    08 AD   7D   008C   397         MOVQ    8(FP),-(SP)             ; PUSH AP/FP
                      0FF0 8F   BB   0090   398         PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>; SAVE R4-R11
                   7E    14 AD   7D   0094   399         MOVQ    20(FP),-(SP)            ; Push R2 and R3
                   50    08 BC   DE   0098   400         MOVAL   @8(AP),R0               ; POINT TO MECHANISM ARGS
                   7E    0C A0   7D   009C   401         MOVQ    12(R0),-(SP)            ; SAVE R0-R1
                                      00A0   402
                                      00A0   403         Set_CmdFlg                      ; Set command mode                      [18]
                                      00A8   404         Set_Except                      ; Set Exception bit                     [18]
                                      00B0   405
          0000FE40'EF    01   D0   00B0   406         MOVL    #APM$_SYSERR, -
                                      00B7   407                 L^DSA$GL_MSGTYP         ; Inform APT of error
                       FF46'   30   00B7   408         BSBW    SCRIPT$STOP             ; GET OUT OF SCRIPT MODE
          00000000'EF    6E   FA   00BA   409         CALLG   (SP),L^DS$CLI           ; PAUSE FOR OPERATOR                    [14]
                                      00C1   410
                50    04 AC   7D   00C1   411         MOVQ    4(AP),R0                ; POINT TO SIGNAL/MECHANISM ARRAY
```

```
                     OC A1    8E    7D  00C5    412          MOVQ    (SP)+,12(R1)                ; PUT MODIFIED R0/R1 BACK IN MECHANISM
                        OFFC  8F    BA  00C9    413          POPR    #^XFFC                      ; RESTORE R2-R11
                                        00CD    414
            00000004'EF      8E    7D  00CD    415          MOVQ    (SP)+,L^NEW_AP_FP           ; AP - FP                                   [14]
            00000000'EF      8E    D0  00D4    416          MOVL    (SP)+,L^NEW_SP              ; PUT SP, PC, PSL                           [14]
            0000000C'EF      8E    7D  00DB    417          MOVQ    (SP)+,L^NEW_PC_PSL          ; IN A SAFE PLACE                           [14]
                        51    60    D0  00E2    418          MOVL    (R0),R1                     ; SIGNAL ARRAY SIZE
                           6041    DC  00E5    419          MOVPSL  (R0)[R1]                    ; REPLACE ORIGINAL PSL WITH CURRENT
                     FC A041    F2'AF  DE  00E8    420          MOVAL   B^20$,-4(R0)[R1]            ; LOCAL PC REPLACES ORIGINAL
                        50    01    D0  00EE    421          MOVL    #SS$_CONTINUE,R0            ; INDICATE SUCCESS
                                    04  00F1    422          RET                                 ; RETURN TO DISPATCHER
                                        00F2    423
            5C  00000004'EF      7D  00F2    424 20$:     MOVQ    L^NEW_AP_FP,AP              ; SET AP AND SP                             [14]
            5E  00000000'EF      D0  00F9    425          MOVL    L^NEW_SP,SP                 ; SWITCH "SP'S"                             [14]
            7E  0000000C'EF      7D  0100    426          MOVQ    L^NEW_PC_PSL,-(SP)          ; SETUP FOR REI  PC/PSL PAIR                [14]
                                    02  0107    427          REI                                 ; RETURN TO PROGRAM CAUSING EXCEPTION
```

```
                                        .SBTTL  DSX$PrintSig Routine - Format Signal Array
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine can be called to format the signal array
                                ;       created by the condition/exception handler.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       DSX$PRINTSIG(signal_array)      Print using DS$PRINTB/DS$PRINTX
                                ;       DSX$PRINTSIGI(signal_array)     Print using DS$PRINTI
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       AP -> Signal array, Specific contents dependent on the condition
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       NONE
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       NONE
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       An error message is typed
                                ;
                                ; REGISTER USAGE:
                                ;
                                ;       R5      Print routine for message header
                                ;       R6      Print routine for information text
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       SS$_NORMAL      If condition recoginzed
                                ;       SS$_RESIGNAL    If condition unknown
                                ;--
                                
                                        $OFFSET 0,NEGATIVE, < -
                                                <BUFFER,BUFSIZ> -       ; Length of buffer for conversion
                                                <LOCAL,0>>              ; Length of local storage
FF00                            BUFFER:
FF00                            LOCAL:
```

```
                     006C  0108    476 .ENTRY  DSX$PRINTSIG,^M<R2,R3,R5,R6>    ; Save registers
55   00000000'9F       9E  010A    477         MOVAB   a#DSX$PRINTB,R5         ; Use PRINTB for first message
56   00000000'9F       9E  0111    478         MOVAB   a#DSX$PRINTX,R6         ; Use PRINTX for information message
                       0C  11  0118    479         BRB     COMMON                  ; Join common code
                           011A    480
                     006C  011A    481 .ENTRY  DSX$PRINTSIGI,^M<R2,R3,R5,R6>; Save registers
55   00000000'9F       9E  011C    482         MOVAB   a#DSX$PRINTI,R5         ; Use PRINTI for first message
          56   65     9E  0123    483         MOVAB   (R5),R6                 ; Use PRINTI for information message
                           0126    484
                           0126    485 COMMON:
00000000'EF       0B    90  0126    486         MovB    #DS$K_Type_Exception_Head, - ; Set typecode                 [18]
                           012D    487           L^DS$GB_TypeCode      ;                                           [18]
                           012D    488
    5E   FF00 CD     9E  012D    489         MOVAB   LOCAL(FP),SP            ; Allocate space for local storage
 50   04 AC   08     C7  0132    490         DIVL3   #8,4(AP),R0             ; Get index from condition code
                           0137    491
     00000360'EF       9F  0137    492         PUSHAB  L^PRINT_PC_PSL          ; Fake JSB to handler:                  [18]
                           013D    493                                         ; ... set return address to print PC/PSL[14]
                           013D    494         CASE    R0,LIMIT=#SS$_BREAK@-3,TYPE=L,DISPLIST=<-
                           013D    495           X_BREAK, -                    ; SS$_BREAK
                           013D    496           X_CHMS, -                     ; SS$_CHMS
                           013D    497           X_CHMU, -                     ; SS$_CHMU
                           013D    498           X_COMPAT, -                   ; SS$_COMPAT
                           013D    499           X_OPCCUS, -                   ; SS$_OPCCUS
                           013D    500           X_OPCDEC, -                   ; SS$_OPCDEC
                           013D    501           RESIGNAL, -                   ; SS$_PAGRDERR, not handled
                           013D    502           X_RADRMOD, -                  ; SS$_RADRMOD
                           013D    503           X_ROPRAND, -                  ; SS$_ROPRAND
                           013D    504           RESIGNAL, -                   ; SS$_FAIL, not handled
                           013D    505           X_TBIT, -                     ; SS$_TBIT
                           013D    506           RESIGNAL, -                   ; SS$_DEBUG, not handled
                           013D    507           RESIGNAL, -                   ; SS$_ARTRES, not handled
                           013D    508           X_INTOVF, -                   ; SS$_INTOVF
                           013D    509           X_INTDIV, -                   ; SS$_INTDIV
                           013D    510           X_FLTOVF, -                   ; SS$_FLTOVF
                           013D    511           X_FLTDIV, -                   ; SS$_FLTDIV
                           013D    512           X_FLTUND, -                   ; SS$_FLTUND
                           013D    513           X_DECOVF, -                   ; SS$_DECOVF
                           013D    514           X_SUBRNG, -                   ; SS$_SUBRNG
                           013D    515           X_FLTOVF_F, -                 ; SS$_?????? Floating overflow fault
                           013D    516           X_FLTDIV_F, -                 ; SS$_?????? Floating divide fault
                           013D    517           X_FLTUND_F -                  ; SS$_?????? Floating underflow fault
                           013D    518           >
                           0173    519
     04 AC   0C    B1  0173    520         CMPW    #SS$_ACCVIO,4(AP)       ; Access violation?
                 0A    12  0177    521         BNEQ    200$                    ; Branch if not
   65   000000B7'EF    FA  0179    522         CALLG   L^MSG_ACCVIO,(R5)       ; Print header                  [14]
              01B1    31  0180    523         BRW     PRINT_TYPE_VA           ; Format access type and address
                           0183    524
04 AC   000002BC 8F    D1  0183    525 200$:   CmpL    #SS$_MCHECK, 4(Ap)      ; User mode machine check?      [17]
                 08    12  018B    526         BNeq    205$                    ; Branch if not                 [17]
   65   00000037'EF    FA  018D    527         CallG   L^Msg_UMChk, (R5)        ; Print header                  [17]
                       05  0194    528         Rsb                             ; Print PC/PSL                  [17]
                           0195    529
04 AC   00660088 8F    D1  0195    530 205$:   CMPL    #DS$_MCHK,4(AP)         ; Standalone Machine check?     [17]
                 0E    12  019D    531         BNEQ    210$                    ; Branch if not
              0060 8F    BB  019F    532         PUSHR   #^M<R5,R6>              ; Use Separate print routines for hdr/txt
```

```
                                5C   DD  01A3   533          PUSHL   AP                          ; Address of signal array
                00000000'EF     03   FB  01A5   534          CALLS   #3,CHR$MCHK                 ; Format Machine check information
                                05   01AC  535               RSB                                 ; All done
                                     01AD  536
    04 AC       00660090 8F     D1   01AD  537 210$:  CMPL    #DS$_KRNLSTK,4(AP)                  ; Kernel stack not valid?
                             08     12   01B5  538          BNEQ    220$                        ; Branch if not
                    65   0000003F'EF     FA   01B7  539          CALLG   L^MSG_KRNLSTK,(R5)          ; Print header                          [14]
                                05   01BE  540               RSB                                 ; All done
                                     01BF  541
    04 AC       006600A0 8F     D1   01BF  542 220$:  CMPL    #DS$_TRANSL,4(AP)                   ; Translation not valid?
                             0A     12   01C7  543          BNEQ    240$                        ; Branch if not                         [23]
                    65   000000CB'EF     FA   01C9  544          CALLG   L^MSG_TRANSL,(R5)           ; Print header                          [14]
                             0161   31   01D0  545          BRW     PRINT_TYPE_VA               ; Format access type and address
                                     01D3  546
    04 AC       006600A8 8F     D1   01D3  547 240$:  CMPL    #DS$_CHME,4(AP)                     ; Change mode to EXEC?
                             09     12   01DB  548          BNEQ    250$                        ; Branch if not
                    65   0000012F'EF     FA   01DD  549          CALLG   L^MSG_CHME,(R5)             ; Print header                          [14]
                             11     11   01E4  550          BRB     255$                        ; Branch to print argument
                                     01F6  551
    04 AC       006600E0 8F     D1   01E6  552 250$:  CMPL    #DS$_CHMK,4(AP)                     ; Change mode to KERNEL?
                             0A     12   01EE  553          BNEQ    260$                        ; Branch if not
                    65   0000011B'EF     FA   01F0  554          CALLG   L^MSG_CHMK,(R5)             ; Print header                          [14]
                             0151   31   01F7  555 255$:  BRW     PRINT_CHMX_ARG              ; Print argument
                                     01FA  556
    04 AC       006600D8 8F     D1   01FA  557 260$:  CMPL    #DS$_UNEXPINT,4(AP)                 ; Unexpected trap or interrupt?
                             0D     12   0202  558          BNEQ    270$                        ; Branch if not
                        08 AC       DD   0204  559          PUSHL   8(AP)                       ; Vector offset
                000003E7'EF         9F   0207  560          PUSHAB  L^FMT_UNEXPINT              ; Edit string                           [14]
                    65   02         FB   020D  561          CALLS   #2,(R5)                     ; Print it
                                05   0210  562               RSB                                 ; All done
                                     0211  563
    04 AC       006600D0 8F     D1   0211  564 270$:  CMPL    #DS$_ARITH,4(AP)                    ; Arithmetic fault or trap?
                             1F     12   0219  565          BNEQ    280$                        ; Branch if this not it
                    0360'CF         9F   021B  566          PUSHAB  W^PRINT_PC_PSL              ; Set return address to print PC/PSL
                                     021F  567          CASE    8(AP),TYPE=W,LIMIT=#1,DISPLIST=<-       ; Case on Trap/Fault type
                                     021F  568                    X_INTOVF, -                 ; SS$_INTOVF
                                     021F  569                    X_INTDIV, -                 ; SS$_INTDIV
                                     021F  570                    X_FLTOVF, -                 ; SS$_FLTOVF
                                     021F  571                    X_FLTDIV, -                 ; SS$_FLTDIV
                                     021F  572                    X_FLTUND, -                 ; SS$_FLTUND
                                     021F  573                    X_DECOVF, -                 ; SS$_DECOVF
                                     021F  574                    X_SUBRNG, -                 ; SS$_SUBRNG
                                     021F  575                    X_FLTOVF_F, -               ; SS$_?????? Floating overflow fault
                                     021F  576                    X_FLTDIV_F, -               ; SS$_?????? Floating divide fault
                                     021F  577                    X_FLTUND_F -                ; SS$_?????? Floating underflow fault
                                     021F  578                    >
                             00     11   0238  579          BRB     PRINT_UNK_EXC               ; Bad value of trap code, fake it.
                                     023A  580
                                     023A  581 ;+
                                     023A  582 ; Unknown exception code, Print the arglist and PC/PSL
                                     023A  583 ;-
                                     023A  584
                                     023A  585 280$:
                                     023A  586 PRINT_UNK_EXC:
            00000000'EF     0B   90   023A  587          MOVB    #DS$K_Type_Exception_Head, - ; Set typecode for header          [18]
                                     0241  588                    L^DS$GB_TypeCode    ;       [18]
                                     0241  589
```

```
                000004F0'EF   9F   0241    590          PUSHAB  L^FMT_UNK_ARG           ; Edit string for count                 [14]
                    65   02   FB   0247    591          CALLS   #2,(R5)                 ; Print header
                                  024A    592
          00000000'EF    0C   90   024A    593          MovB    #DS$K_Type_Exception, - ; Set typecode for text                 [18]
                                  0251    594            L^DS$GB_TypeCode              ;                                       [18]
                         6C   DD   0251    595          PUSHL   (AP)                    ; Count of args
                00000512'EF   9F   0253    596          PUSHAB  L^FMT_UNK_CNT           ; format for count of args              [14]
                    66   02   FB   0259    597          CALLS   #2,(R6)                 ; Print count of args
                                  025C    598
               53   6C   02   C3   025C    599          SUBL3   #2,(AP),R3              ; Count less PC/PSL
                         15   15   0260    600          BLEQ    20$                     ; None, Print PC/PSL
                    52   01   D0   0262    601          MOVL    #1,R2                   ; Start with first arg
                                  0265    602
                      6C42   DD   0265    603 10$:     PUSHL   (AP)[R2]                ; Value of arg
                         52   DD   0268    604          PUSHL   R2                      ; arg number
                00000527'EF   9F   026A    605          PUSHAB  L^FMT_PXXX              ; Address of edit string                [14]
                    66   03   FB   0270    606          CALLS   #3,(R6)                 ; Print arg
                 EE 52   53   F2   0273    607          AOBLSS  R3,R2,10$               ; Print all args
                                  0277    608
                      00E6   31   0277    609 20$:     BRW     PRINT_PC_PSL            ; Now format and print PC/PSL
```

```
                                027A   611 ;+
                                027A   612 ; SS$_BREAK              Breakpoint exception
                                027A   613 ;-
                                027A   614
                                027A   615 X_BREAK:
      65    000000F3'EF     FA 027A   616          CALLG   L^MSG_BREAK,(R5)         ; Print header                          [14]
                            05 0281   617          RSB                              ; All done
                               0282   618
                               0282   619 ;+
                               0282   620 ; SS$_CHMS               Change mode to SUPERVISOR
                               0282   621 ;-
                               0282   622
                               0282   623 X_CHMS:
      65    00000143'EF     FA 0282   624          CALLG   L^MSG_CHMS,(R5)          ; Print header                          [14]
                            05 0289   625          RSB                              ; All done
                               028A   626
                               028A   627 ;+
                               028A   628 ; SS$_CHMU               Change mode to USER
                               028A   629 ;-
                               028A   630
                               028A   631 X_CHMU:
      65    00000157'EF     FA 028A   632          CALLG   L^MSG_CHMU,(R5)          ; Print header                          [14]
                            05 0291   633          RSB                              ; All done
                               0292   634
                               0292   635 ;+
                               0292   636 ; SS$_COMPAT             Compatability mode trap
                               0292   637 ;-
                               0292   638
                               0292   639 X_COMPAT:
   65   00000107'EF     FA 0292   640          CALLG   L^MSG_COMPAT,(R5)        ; Print Compatability mode fault header [14]
50  08 AC   03   00    EF 0299   641          EXTZV   #0,#3,8(AP),R0           ; Get fault type
     000002F0'EF40     DD 029F   642          PUSHL   L^AAT_COMPAT[R0]         ; Address of insert                     [14]
       0000069A'EF     9F 02A6   643          PUSHAB  L^T_FLT_ABO              ; Address of FAULT/ABORT                [14]
       000006A6'EF     9F 02AC   644          PUSHAB  L^T_COMPAT               ; Address of Compatability mode ...     [14]
       000004C6'EF     9F 02B2   645          PUSHAB  L^FMT_EXCEP_CODE         ; Format for exception code             [14]
           66   04     FB 02B8   646          CALLS   #4,(R6)                  ; Print type code
                        05 02BB   647          RSB                              ; All done
                               02BC   648
                               02BC   649 ;+
                               02BC   650 ; SS$_OPCCUS              Opcode reserved to customer
                               02BC   651 ;-
                               02BC   652
                               02BC   653 X_OPCCUS:
      65    0000007B'EF     FA 02BC   654          CALLG   L^MSG_OPCCUS,(R5)        ; Print header                          [14]
                            05 02C3   655          RSB                              ; All done
                               02C4   656
                               02C4   657 ;+
                               02C4   658 ; SS$_OPCDEC              Opcode reserved to DIGITAL
                               02C4   659 ;-
                               02C4   660
                               02C4   661 X_OPCDEC:
      65    00000067'EF     FA 02C4   662          CALLG   L^MSG_OPCDEC,(R5)        ; Print header                          [14]
                            05 02CB   663          RSB                              ; All done
                               02CC   664
                               02CC   665 ;+
                               02CC   666 ; SS$_RADRMOD             Reserved addressing mode
                               02CC   667 ;-
```

```
                        02CC    668
                        02CC    669 X_RADRMOD:
65  000000A3'EF     FA  02CC    670         CALLG   L^MSG_RADRMOD,(R5)      ; Print header                  [14]
                    05  02D3    671         RSB                             ; All done
                        02D4    672
                        02D4    673 ;+
                        02D4    674 ; SS$_ROPRAND           Reserved operand
                        02D4    675 ;-
                        02D4    676
                        02D4    677 X_ROPRAND:
65  0000008F'EF     FA  02D4    678         CALLG   L^MSG_ROPRAND,(R5)      ; Print header                  [14]
                    05  02D8    679         RSB                             ; All done
                        02DC    680
                        02DC    681 ;+
                        C2DC    682 ; SS$_TBIT              T-bit exception
                        02DC    683 ;-
                        02DC    684
                        02DC    685 X_TBIT:
65  000000DF'EF     FA  02DC    686         CALLG   L^MSG_TBIT,(R5)         ; Print header                  [14]
                    05  02E3    687         RSB                             ; All done
                        02E4    688
                        02E4    689 ;+
                        02E4    690 ; SS$_INTOVF            Integer overflow
                        02E4    691 ;-
                        02E4    692
                        02E4    693 X_INTOVF:
65  00000228'EF     FA  02E4    694         CALLG   L^MSG_INTOVF,(R5)       ; Print header                  [14]
                    05  02EB    695         RSB                             ; All done
                        02EC    696 ;+
                        02EC    697 ; SS$_INTDIV            Integer division by zero
                        02EC    698 ;-
                        02EC    699
                        02EC    700 X_INTDIV:
65  0000023C'EF     FA  02EC    701         CALLG   L^MSG_INTDIV,(R5)       ; Print header                  [14]
                    05  02F3    702         RSB                             ; All done
                        02F4    703
                        02F4    704 ;+
                        02F4    705 ; SS$_FLTOVF            Floating overflow
                        02F4    706 ;-
                        02F4    707
                        02F4    708 X_FLTOVF:
65  00000250'EF     FA  02F4    709         CALLG   L^MSG_FLTOVF,(R5)       ; Print header                  [14]
                    05  02FB    710         RSB                             ; All done
                        02FC    711
                        02FC    712 ;+
                        02FC    713 ; SS$_FLTDIV            Floating division by zero
                        02FC    714 ;-
                        02FC    715
                        02FC    716 X_FLTDIV:
65  00000264'EF     FA  02FC    717 170$:   CALLG   L^MSG_FLTDIV,(R5)       ; Print header                  [14]
                    05  0303    718         RSB                             ; All done
                        0304    719
                        0304    720 ;+
                        0304    721 ; SS$_FLTUND            Floating underflow trap
                        0304    722 ;-
                        0304    723
                        0304    724 X_FLTUND:
```

```
                                 0304   725          CALLG   L^MSG_FLTUND,(R5)       ; Print header                          [14]
                          05     030B   726          RSB                             ; All done
                                 030C   727
                                 030C   728 ;+
                                 030C   729 ; SS$_DECOVF            Decimal overflow trap
                                 030C   730 ;-
                                 030C   731
                                 030C   732 X_DECOVF:
                                 030C   733          CALLG   L^MSG_DECOVF,(R5)       ; Print header                          [14]
                          05     0313   734          RSB                             ; All done
                                 0314   735
                                 0314   736 ;+
                                 0314   737 ; SS$_SUBRNG            Subscript range trap
                                 0314   738 ;-
                                 0314   739
                                 0314   740 X_SUBRNG:
                                 0314   741          CALLG   L^MSG_SUBRNG,(R5)       ; Print header                          [14]
                          05     031B   742          RSB                             ; All done
                                 031C   743
                                 031C   744 ;+
                                 031C   745 ; SS$_??????            Floating overflow fault
                                 031C   746 ;-
                                 031C   747
                                 031C   748 X_FLTOVF_F:
                                 031C   749          CALLG   L^MSG_FLTOVF_F,(R5)     ; Print header                          [14]
                          05     0323   750          RSB                             ; All done
                                 0324   751
                                 0324   752 ;+
                                 0324   753 ; SS$_??????            Floating division by zero fault
                                 0324   754 ;-
                                 0324   755
                                 0324   756 X_FLTDIV_F:
                                 0324   757          CALLG   L^MSG_FLTDIV_F,(R5)     ; Print header                          [14]
                                 032B   758
                          05     032B   759          RSB                             ; All done
                                 032C   760
                                 032C   761 ;+
                                 032C   762 ; SS$_??????            Floating underflow fault
                                 032C   763 ;-
                                 032C   764
                                 032C   765 X_FLTUND_F:
                                 032C   766          CALLG   L^MSG_FLTUND_F,(R5)     ; Print header                          [14]
                          05     0333   767          RSB                             ; All done
                                 0334   768
                                 0334   769 Print_Type_VA:                           ;                                       [18]
00000000'EF       0C     90     0334   770          MOVB    #DS$K_Type_Exception, - ; Set typecode                          [18]
                                 033B   771            L^DS$GB_TypeCode      ;                                       [18]
                                 033B   772
                    08   AC    DD     033B   773          PushL   8(AP)                   ; Reference type                        [18]
                    0C   AC    DD     033E   774          PUSHL   12(AP)                  ; Virtual address
          00000494'EF       9F     0341   775          PUSHAB  L^FMT_VIRT              ; Format string                         [14]
                66   03     FB     0347   776          CALLS   #3,(R6)                 ; Print message of lesser importance
                          05     034A   777          RSB                             ; Now format and print PC/PSL
                                 034B   778
                                 034B   779 PRINT_CHMX_ARG:
00000000'EF       0C     90     034B   780          MOVB    #DS$K_Type_Exception, - ; Set typecode                          [18]
                                 0352   781            L^DS$GB_TypeCode      ;                                       [18]
```

```
                        0352   782
                 08 AC  DD 0352   783          PUSHL   8(AP)                   ; Output CHMx ARG
        000004DD'EF     9F 0355   784          PUSHAB  L^FMT_CHMX_ARG          ; Format CHMx ARG                       [14]
              66   02   FB 035B   785          CALLS   #2,(R6)                 ; Output message of lesser importance
                   00   11 035E   786          BRB     PRINT_PC_PSL            ; Now format and print PC/PSL
                        0360   787
                        0360   788 PRINT_PC_PSL:
00000000'EF       0C    90 0360   789          MovB    #DS$K_Type_Exception, - ; Set typecode                          [18]
                        0367   790            L^DS$GB_TypeCode      ;                                               [18]
                        0367   791
                   52   6C   D0 0367   792          MOVL    (AP),R2                 ; Get count of args
                   FC AC42   DD 036A   793          PUSHL   -4(AP)[R2]              ; PC to print
        0000041F'EF     9F 036E   794          PUSHAB  L^FMT_ERRPC             ; Address of format string              [14]
              66   02   FB 0374   795          CALLS   #2,(R6)                 ; Output message of lesser importance
                        0377   796          $DS_CVTREG_S    #31,(AP)[R2], - ; MSB and value to be converted
                        0377   797                  APSLMNE, BUFFER(FP), -  ; Control string and buffer address
                        0377   798                  #BUFSIZ, MODELST,MODELST; Buffer size and mode list twice
                50 FF00 CD    9E 03A7   799          MOVAB   BUFFER(FP),R0           ; Get address of buffer
                         50   DD 03AC   800          PUSHL   R0                      ; Address of mneumonic string
                       6C42   DD 03AE   801          PUSHL   (AP)[R2]                ; Push PSL
        00000473'EF     9F 03B1   802          PUSHAB  L^FMT_ERRPSL            ; Address of format string              [14]
              66   03   FB 03B7   803          CALLS   #3,(R6)                 ; output message of lesser importance
        00000000'EF     D4 03BA   804          ClrL    L^DS$GB_TypeCode        ; Clear typecode                        [18]
                   04   03C0   805          RET                             ; Return all done
                        03C1   806
                        03C1   807 RESIGNAL:
                50 0918 8F    3C 03C1   808          MOVZWL  #SS$_RESIGNAL,R0        ; Unknown signal
        00000000'EF     D4 03C6   809          ClrL    L^DS$GB_TypeCode        ; Clear typecode                        [18]
                   04   03CC   810          RET                             ; Return
```

```
                        812          .SBTTL  FIND_USERPC     Find USER PC on stack
                        813 ;++
                        814 ; FUNCTIONAL DESCRIPTION:
                        815 ;
                        816 ;       This routine is used to find and print a user program PC
                        817 ;       in the event of an error, exception, abort or fault.
                        818 ;
                        819 ; CALLING SEQUENCE:
                        820 ;
                        821 ;       CALLG   (AP),FIND_USERPC
                        822 ;
                        823 ; INPUT PARAMETERS:
                        824 ;
                        825 ;       R0 = ADDRESS OF FAILURE
                        826 ;
                        827 ; IMPLICIT INPUTS:
                        828 ;
                        829 ;       NONE
                        830 ;
                        831 ; OUTPUT PARAMETERS:
                        832 ;
                        833 ;       NONE
                        834 ;
                        835 ; IMPLICIT OUTPUTS:
                        836 ;
                        837 ;       NONE
                        838 ;
                        839 ; COMPLETION CODES:
                        840 ;
                        841 ;       NONE
                        842 ;
                        843 ; SIDE EFFECTS:
                        844 ;
                        845 ;       NONE
                        846 ;--
```

```
                    0000  03CD   848 .ENTRY FIND_USERPC,^M<>
                          03CF   849
          50   04 BC    DE  03CF   850          MOVAL   @4(AP),R0               ; GET ADDRESS OF SIGNAL ARRAY
             51   60    9A  03D3   851          MOVZBL  (R0),R1                 ; GET COUNT OF SIGNAL ARGS
       50    FC A041    D0  03D6   852          MOVL    -4(R0)[R1],R0           ; Get PC of failure
00010000 8F     50    D1  03DB   853          cmpl    r0, #^X10000            ; Less than 10000?                      [15]
                0C    1F  03E2   854          blssu   10$                     ; If so, don't search                   [15]
00000000'EF     00    FB  03E4   855          CallS   #0, L^DSR$Find_User_PC  ; Search for it                         [20]
             50   00    D1  03EB   856          CmpL    #0, R0                  ; Return a zero?                        [20]
                10    13  03EE   857          Beql    20$                     ; Yes, so no user PC was found          [20]
                          03F0   858
                          03F0   859 10$:     $PRINTI_S  L^FMT_USRPC, R0      ; Print the real User PC                [20]
                    04    03FF   860          RET                             ; Return to caller
                          0400   861
                          0400   862 20$:     $PrintI_S  L^Fmt_User_PC_NF     ; Print "none found"                    [20]
                    04    040D   863          RET                             ; Return to caller                      [20]
```

```
                        865          .SBTTL  TST$MChk Routine - Machine Check Handler        ;                [14]
                        866          .PSECT  Code, Exe, Shr, NoWrt, Long                     ;                [14]
                        867 ;++
                        868 ; FUNCTIONAL DESCRIPTION:
                        869 ;
                        870 ;       This routine is entered from SCB offset 4.
                        871 ;       It removes the machine check info from the stack and adds 2
                        872 ;       to the PC, sets the "C" bit in the PSL and returns
                        873 ;
                        874 ; CALLING SEQUENCE:
                        875 ;
                        876 ;       Vectored from SCB offset 4
                        877 ;       FOR EXAMPLE:
                        878 ;               PUSHL   4(scb)          ; Save old machine check
                        879 ;               MOVAL   TST$MCHK,4(SCB) ; Set new machine check handler
                        880 ;               TSTW    (R0)            ; Instruction expected to get mchk
                        881 ;               POPL    4(SCB)          ; Restore old machine check handler
                        882 ;               BCS     ...             ; Branch if machine check occured
                        883 ;
                        884 ; INPUT PARAMETERS:
                        885 ;
                        886 ;       NONE
                        887 ;
                        888 ; IMPLICIT INPUTS:
                        889 ;
                        890 ;       (SP)    number of bytes of machine check info
                        891 ;       (SP)+SP PC and PSL of fault inst which MUST be 2 bytes long
                        892 ;
                        893 ; OUTPUT PARAMETERS:
                        894 ;
                        895 ;       NONE
                        896 ;
                        897 ; IMPLICIT OUTPUTS:
                        898 ;
                        899 ;       C-bit is set and interrupt PC updated by 2
                        900 ;
                        901 ; SIDE EFFECTS:
                        902 ;
                        903 ;       NONE
                        904 ;
                        905 ; COMPLETION CODES:
                        906 ;
                        907 ;       N/A
                        908 ;--
```

```
                          040E   910          .ALIGN  LONG
                          0410   911 TST$MCHK::
                    5E    DD     0410   912          PUSHL   SP                          ; Store stack frame address
                    01    DD     0412   913          PUSHL   #DS$_GK_TST$MCHK                    ; Pass fault clear selection code
00000000'           EF    16     0414   914          JSB     L^DSR$FAULT_CLEAR           ; Clear fault status
                    5E    08     C0     041A   915   ADDL2   #8,SP                       ; Reset the stack                           ;[24
                    5E    8E     C0     041D   916   ADDL    (SP)+, SP                   ; Remove specific machine check info    [16]
                    6E    02     C0     0420   917   ADDL    #2,(SP)                     ; Update PC
                04  AE    01     C8     0423   918   BISL    #1,4(SP)                    ; Set C-bit in condition codes
                                 02     0427   919   REI                                ; Return after faulting instruction
```

```
                     0428   921          .SBTTL  HARDWARE EXCEPTION ENTRY POINTS
                00000428    922          .PSECT  Code, Exe, Shr, NoWrt, Long                     ;                [14]
                     0428   923 ;++
                     0428   924 ; FUNCTIONAL DESCRIPTION:
                     0428   925 ;
                     0428   926 ;       THESE ROUTINES EMULATE THE STARLET EXCEPTION HANDLER MECHANISM.
                     0428   927 ;       SEE THE STARLET WORKING DESIGN DOCUMENT FOR FURTHER DETAILS.
                     0428   928 ;
                     0428   929 ; CALLING SEQUENCE:
                     0428   930 ;
                     0428   931 ;       EXCEPTION OR INTERRUPT.
                     0428   932 ;
                     0428   933 ; INPUT PARAMETERS:
                     0428   934 ;
                     0428   935 ;       EXCEPTION FRAME ON THE STACK.
                     0428   936 ;
                     0428   937 ; IMPLICIT INPUTS:
                     0428   938 ;
                     0428   939 ;       NONE
                     0428   940 ;
                     0428   941 ; OUTPUT PARAMETERS:
                     0428   942 ;
                     0428   943 ;       NONE
                     0428   944 ;
                     0428   945 ; IMPLICIT OUTPUTS:
                     0428   946 ;
                     0428   947 ;       NONE
                     0428   948 ;
                     0428   949 ; COMPLETION CODES:
                     0428   950 ;
                     0428   951 ;       NONE
                     0428   952 ;
                     0428   953 ; SIDE EFFECTS:
                     0428   954 ;
                     0428   955 ;       NONE
                     0428   956 ;
                     0428   957 ;--
```

```
                     0428   959 ;+
                     0428   960 ; ENTER HERE FOR MACHINE CHECK ABORTS, FAULTS AND TRAPS
                     0428   961 ;-
                     0428   962
                     0428   963          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0428   964 EXE_MCHK::                                   ; HERE FROM MCHK VECTOR
 00660088 8F    DD   0428   965          PUSHL   #DS$_MCHK                   ; INDICATE MACHINE CHECK CONDITION
7E    04 AE    04   C7   042E   966          DIVL3   #4,4(SP),-(SP)              ; CONVERT BYTES TO LONGWORD COUNT
      6E    04   C0   0433   967          ADDL2   #4,(SP)                     ; INCLUDE PSL/PC/COND NAME & BYTE COUNT
            5E   DD   0436   968          PUSHL   SP                          ; Store stack frame address                 ;C24
            00   DD   0438   969          PUSHL   #DS$_GK_EXE_MCHK            ; PUSH THE FAULT CLEAR SELECTION CODE       ;C24
 00000000'EF    16   043A   970          JSB     L^DSR$FAULT_CLEAR           ; CLEAR ERROR FIRST PASS AND RE-INIT
      5E    08   C0   0440   971          ADDL2   #8,SP                       ; Reset the stack                           ;C24
          00E1   31   0443   972          BRW     EXE_EXCEPTION               ; PROCEED
                     0446   973
                     0446   974 ;+
                     0446   975 ; ENTER HERE FOR KERNEL STACK NOT VALID ABORT
                     0446   976 ;-
                     0446   977
                     0446   978          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0448   979 EXE_KRNLSTK::                                ; HERE FROM KRNLSTK VECTOR
 00660090 8F    DD   0448   980          PUSHL   #DS$_KRNLSTK                ; INDICATE KERNEL STACK CONDITION
          00D2   31   044F   981          BRW     EXE_3ARG                    ; SETUP 3 ARGS, AND PROCESS
                     0452   982
                     0452   983 ;+
                     0452   984 ; ENTER HERE FOR POWER INTERRUPT
                     0452   985 ;-
                     0451   986
                     0451   987          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0454   988 EXE_POWER::                                   ; HERE FROM POWER FAIL VECTOR
 00660098 8F    DD   0454   989          PUSHL   #DS$_POWER                  ; INDICATE POWER FAIL CONDITION
            00   00   045A   990          HALT                                ; NOT GRACEFUL, BUT EFFECTIVE
          00C5   31   045B   991          BRW     EXE_3ARG                    ; SETUP 3 ARGS, AND PROCESS
                     045E   992
                     045E   993 ;+
                     045E   994 ; ENTER HERE FOR RESERVED OR PRIVILEGED INSTRUCTION FAULT
                     045E   995 ;-
                     045E   996
                     045E   997          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0460   998 EXE_OPCDEC::                                 ; HERE FROM OPCDEC VECTOR
7E   043C 8F    3C   0460   999          MOVZWL  #SS$_OPCDEC, -(SP)          ; INDICATE RESERVED OPCODE CONDITION
          00BB   31   0465  1000          BRW     EXE_3ARG                    ; SETUP 3 ARGS
                     0468  1001
                     0468  1002 ;+
                     0468  1003 ; ENTER HERE FOR CUSTOMER RESERVED INSTRUCTION FAULT
                     0468  1004 ;-
                     0468  1005
                     0468  1006          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0468  1007 EXE_OPCCUS::                                 ; HERE FROM OPCCUS VECTOR
7E   0434 8F    3C   0468  1008          MOVZWL  #SS$_OPCCUS, -(SP)          ; INDICATE CUSTOMER OPCODE CONDITION
          00B3   31   046D  1009          BRW     EXE_3ARG                    ; SETUP 3 ARGS
```

```
                     0470  1011 ;+
                     0470  1012 ; ENTER HERE FOR RESERVED OPERAND FAULT OR ABORT
                     0470  1013 ;-
                     0470  1014
                     0470  1015          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0470  1016 EXE_ROPRAND::                                ; HERE FROM ROPRAND VECTOR
                     0470  1017 EXE$ROPRAND::                                ; Used by XDELTA only
 7E  0454 8F   3C    0470  1018          MOVZWL  #SS$_ROPRAND, -(SP)         ; INDICATE RESERVED OPERAND CONDITION
     00AB      31    0475  1019          BRW     EXE_3ARG                    ; SETUP 3 ARGS
                     0478  1020
                     0478  1021 ;+
                     0478  1022 ; ENTER HERE FOR RESERVED ADDRESSING MODE FAULT
                     0478  1023 ;-
                     0478  1024
                     0478  1025          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0478  1026 EXE_RADRMOD::                                ; HERE FROM RADRMOD VECTOR
 7E  044C 8F   3C    0478  1027          MOVZWL  #SS$_RADRMOD, -(SP)         ; INDICATE ADDRESSING MODE CONDITION
     00A3      31    047D  1028          BRW     EXE_3ARG                    ; SETUP 3 ARGS
                     0480  1029
                     0480  1030 ;+
                     0480  1031 ; ENTER HERE FOR ACCESS CONTROL VIOLATION FAULT
                     0480  1032 ;-
                     0480  1033
                     0480  1034          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0480  1035 EXE_ACCVIO::                                 ; HERE FROM ACCVIO VECTOR
                     0480  1036 EXE$ACVIOLAT::                               ; Used by XDELTA only
 7E    0C    3C    0480  1037          MOVZWL  #SS$_ACCVIO, -(SP)          ; INDICATE ACCESS VIOLATION CONDITION
       05    DD    0483  1038          PUSHL   #5                          ; INDICATE SIGNAL ARRAY SIZE
     009F    31    0485  1039          BRW     EXE_EXCEPTION               ; PROCEED
                     0488  1040
                     0488  1041 ;+
                     0488  1042 ; ENTER HERE FOR TRANSLATION NOT VALID FAULT
                     0488  1043 ;-
                     0488  1044
                     0488  1045          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0488  1046 EXE_TRANSL::                                 ; HERE FROM TRANSL VECTOR
                     0488  1047 MMG$PAGEFAULT::                              ; Used by XDELTA only
006600A0 8F   DD     0488  1048          PUSHL   #DS$_TRANSL                 ; INDICATE TRANSLATION CONDITION
       05    DD    048E  1049          PUSHL   #5                          ; INDICATE SIGNAL ARRAY SIZE
     0094    31    0490  1050          BRW     EXE_EXCEPTION               ; PROCEED
                     0493  1051
                     0493  1052 ;+
                     0493  1053 ; ENTER HERE FOR COMPATIBILITY FAULT OR ABORT
                     0493  1054 ;-
                     0493  1055
                     0493  1056          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                     0494  1057 EXE_COMPAT::                                 ; HERE FROM COMPAT VECTOR
 7E  042C 8F   3C    0494  1058          MOVZWL  #SS$_COMPAT, -(SP)          ; INDICATE COMPATIBILITY CONDITION
     0083      31    0499  1059          BRW     EXE_4ARG                    ; SETUP 4 ARGS
```

```
                                049C  1061 ;+
                                049C  1062 ; ENTER HERE FOR ARITHMETIC TRAP
                                049C  1063 ;-
                                049C  1064
                                049C  1065          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                                049C  1066 EXE_ARITH::                                  ; HERE FROM ARITH VECTOR
                006600D0 8F     DD    049C  1067          PUSHL   #DS$_ARITH              ; Indicate Arithmetic fault
                         007A   31    04A2  1068          BRW     EXE_4ARG                ; Form 4 argument list
                                04A5  1069
                                04A5  1070 ;+
                                04A5  1071 ; ENTER HERE FOR "ALL" BREAKPOINT FAULTS (I.E., DIAGNOSTIC PROGRAM
                                04A5  1072 ; WILL ALWAYS GET SECOND CHANCE AT THIS VECTOR)
                                04A5  1073 ;-
                                04A5  1074
                                04A5  1075          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                                04A8  1076 EXE_BREAK::                                  ; HERE FROM BREAKPOINT VECTOR
                                04A8  1077 EXE$BREAK::                                  ; Used by XDELTA only
7E   00000000'EF    03    CB   04A8  1078          BICL3   #3,DS$GA_BREAKVEC,-(SP); USE SOFT VECTOR
                     18    13   04B0  1079          BEQL    30$                         ; BRANCH IF NO SOFT VECTOR
                                04B2  1080
                     01    BB   04B2  1081          PUSHR   #^M<R0>                     ; NEED A LITTLE WORKING ROOM
                  50 00'   D0   04B4  1082          MOVL    S^#BPTMAX,R0                ; INIT AN INDEX
00000000'EF40   08 AE    D1   04B7  1083 10$:      CMPL    8(SP),DS$AA_BPTADDR[R0]     ; LOOK AT A SUPERVISOR'S KNOWN BPT'S
                     06    13   04C0  1084          BEQL    20$                         ; IT MATCHES, SO GO CALL HANDLER
                  F2 50    F4   04C2  1085          SOBGEQ  R0,10$                      ; LOOP THRU ENTIRE LIST (INCL. [0])
                     01    BA   04C5  1086          POPR    #^M<R0>                     ; RESTORE R0
                     05    05   04C7  1087          RSB                                 ; ENTER USER I.S.R
                                04C8  1088
                     01    BA   04C8  1089 20$:      POPR    #^M<R0>                     ; RESTORE REG
                6E   0414 8F    3C   04CA  1090 30$:      MOVZWL  #SS$_BREAK, (SP)            ; INDICATE BREAKPOINT CONDITION
                     52    11   04CF  1091          BRB     EXE_3ARG                    ; SETUP 3 ARGS AND DISPATCH
                                04D1  1092
                                04D1  1093
                                04D1  1094 ;+
                                04D1  1095 ; ENTER HERE FOR "ALL" TBIT TRAPS (I.E., DIAGNOSTIC PROGRAM WILL ALWAYS
                                04D1  1096 ; GET SECOND CHANCE AT THIS VECTOR)
                                04D1  1097 ;-
                                04D1  1098
                                04D1  1099          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                                04D4  1100 EXE_TBIT::                                   ; HERE FROM TBIT VECTOR
                                04D4  1101 EXE$TBIT::                                   ; Used by XDELTA only
7E   00000000'EF    03    CB   04D4  1102          BICL3   #3,DS$GA_TBITVEC,-(SP)      ; Set secondary vector
                     08    13   04DC  1103          BEQL    10$                         ; Branch if not set vec'ed
                01 00000000'EF   E8   04DE  1104          BLBS    DEBUG$GB_FLAGS,10$          ; Branch if supervisor expects this
                     05    05   04E5  1105          RSB                                 ; Dispatch to user ISR
                                04E6  1106
                6E   07    D0   04E6  1107 10$:      MOVL    #7,(SP)                     ; Assume compatability mode T-bit
             A6 08 AE    1F   E0   04E9  1108          BBS     #PSL$V_CM,8(SP), -
                                04EE  1109                  EXE_COMPAT                  ; BRANCH TO COMPATABILITY MODE HANDLER
                6E   0464 8F    3C   04EE  1110          MOVZWL  #SS$_TBIT, (SP)             ; INDICATE T-BIT CONDITION
                     2E    11   04F3  1111          BRB     EXE_3ARG                    ; SETUP 3 ARGS, AND DISPATCH
```

```
                     04F5  1113 ;+
                     04F5  1114 ; HERE FOR UNEXPECTED INTERRUPT FROM SCB
                     04F5  1115 ;-
                     04F5  1116 EXE_UNEXPINT::
006600D8 8F    DD    04F5  1117          PUSHL   #DS$_UNEXPINT           ; CODE FOR EXCEPTION ARGLIST
        0021   31    04FB  1118          BRW     EXE_4ARG                ; USE COMMON CODE FOR REST
                     04FE  1119
                     04FE  1120 ;+
                     04FE  1121 ; ENTER HERE FOR CHMK TRAP
                     04FE  1122 ;-
                     04FE  1123
                     04FE  1124          .ALIGN  LONG
                     0500  1125 EXE_CHMK::
006600E0 8F    DD    0500  1126          PUSHL   #DS$_CHMK               ; Indicate extraneous CHMK TRAP
        17     11    0506  1127          BRB     EXE_4ARG                ; And build rest of 4 ARG list
                     0508  1128 ;+
                     0508  1129 ; ENTER HERE FOR CHME TRAP
                     0508  1130 ;-
                     0508  1131
                     0508  1132          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                     0508  1133 EXE_CHME::                               ; HERE FROM CHME VECTOR
006600A8 8F    DD    0508  1134          PUSHL   #DS$_CHME               ; INDICATE CHANGE MODE CONDITION
        0F     11    050E  1135          BRB     EXE_4ARG                ; SETUP 4 ARGS, NO SECONDARY VECTOR
                     0510  1136
                     0510  1137 ;+
                     0510  1138 ; ENTER HERE FOR CHMS TRAP
                     0510  1139 ;-
                     0510  1140
                     0510  1141          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                     0510  1142 EXE_CHMS::                               ; HERE FROM CHMS VECTOR
7E   041C 8F    3C   0510  1143          MOVZWL  #SS$_CMODSUPR, -(SP)    ; INDICATE CHANGE MODE CONDITION
        08     11    0515  1144          BRB     EXE_4ARG                ; SETUP 4 ARGS, NO SECONDARY VECTOR
                     0517  1145
                     0517  1146 ;+
                     0517  1147 ; ENTER HERE FOR CHMU TRAP
                     0517  1148 ;-
                     0517  1149
                     0517  1150          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                     0518  1151 EXE_CHMU::                               ; HERE FROM CHMU VECTOR
7E   0424 8F    3C   0518  1152          MOVZWL  #SS$_CMODUSER, -(SP)    ; INDICATE CHANGE MODE CONDITION
        00     11    051D  1153          BRB     EXE_4ARG                ; SETUP 4 ARGS, NO SECONDARY VECTOR
                     051F  1154
                     051F  1155 EXE_4ARG:
        04     DD    051F  1156          PUSHL   #4                      ; 4 ARGS
        04     11    0521  1157          BRB     EXE_EXCEPTION           ; JOIN COMMON CODE
                     0523  1158
                     0523  1159 EXE_3ARG:
        03     DD    0523  1160          PUSHL   #3                      ; 3 ARGS
        00     11    0525  1161          BRB     EXE_EXCEPTION
```

```
                                0527  1163 ;+
                                0527  1164 ; COMMON CODE FOR ALL EXCEPTIONS.
                                0527  1165 ; STACK CONTAINS SIGNAL ARRAY, BUILD MECHANISM ARRARY
                                0527  1166 ; THEN SWITCH TO PREVIOUS MODE STACK UNLESS 'IS' IS SET OR PREVMODE WAS KERNEL
                                0527  1167 ;-
                                0527  1168
                                0527  1169 EXE_EXCEPTION:
                          03    BB  0527  1170         PUSHR   #^M<R0,R1>              ; SAVE VOLATILE REGISTERS
                       7E 03    CE  0529  1171         MNEGL   #3,-(SP)                ; INITIAL FRAME DEPTH
                          5D    DD  052C  1172         PUSHL   FP                      ; INITIAL HANDLER ESTABLISHER FRAME
                          04    DD  052E  1173         PUSHL   #4                      ; MECHANISM ARRAY SIZE
                                0530  1174
                          51    DC  0530  1175         MOVPSL  R1                      ; GET CURRENT PSL
                    6E 51 1A    E0  0532  1176         BBS     #PSL$V_IS,R1,NORMAL     ; INTERRUPT ON INTERRUPT STACK, PROCESS
                50  51 02    16  EF  0536  1177         EXTZV   #PSL$V_PRVMOD, -
                                0538  1178                 #PSL$S_PRVMOD,R1,R0     ; GET PREVIOUS MODE
                          67    13  053B  1179         BEQL    NORMAL                  ; BRANCH IF WAS IN KERNEL MODE, PROCESS
                                053D  1180
                          0C    BB  053D  1181         PUSHR   #^M<R2,R3>              ; SAVE WORKING REGISTERS
                       52 50    D0  053F  1182         MOVL    R0,R2                   ; SAVE PREVIOUS MODE STACK POINTER
                 53 1C AE 06    C1  0542  1183         ADDL3   #6,28(SP),R3            ; CALCULATE NUMBER OF LONGWORDS TO MOVE
                                0547  1184
             51 00000000'EF42    D0  0547  1185         MOVL    PCB_BASE[R2],R1         ; GET SAVED SP FROM 'PCB'
                       51 52    DB  054F  1186         MFPR    R2,R1                   ; GET HARDWARE SAVED PREV MODE SP
                    50 53 02    78  0552  1187         ASHL    #2,R3,R0                ; CALCULATE NUMBER OF BYTES TO MOVE
                       51 50    C2  0556  1188         SUBL2   R0,R1                   ; SET UP A NEW STACK POINTER
                    61 50 52    0D  0559  1189         PROBEW  R2,R0,(R1)              ; CAN IT BE DONE ?
                          1C    12  055D  1190         BNEQ    50$                     ; 'NO PROBLEM', PROCEED
                                055F  1191                                             ;
                                055F  1192         ERRSUP_S  ,T_STKVIO             ; CANNOT COPY STACK !
                                0574  1193         $DS_ABORT                       ; SO GO AHEAD & DO THE UGLY DEED
                                057B  1194                                             ;
 00000000'EF40    51    D0  057B  1195 50$:    MOVL    R1,PCB_BASE[R0]             ; STORE NEW SP INTO 'PCB'
                 52    51    DA  0583  1196         MTPR    R1,R2                   ; STORE NEW HARDWARE SP
              50 08 AE    9E  0586  1197         MOVAB   8(SP),R0                ; ADDRESS OF ARGS TO COPY
                 81    80    D0  058A  1198 60$:    MOVL    (R0)+,(R1)+             ; COPY CURR STACK TO NEW ONE
                    FA 53    F5  058D  1199         SOBGTR  R3,60$                  ; UNTIL DONE
                                0590  1200
                                0590  1201 ;+
                                0590  1202 ; NOW MAKE THE SWITCH TO THE NEW STACK
                                0590  1203 ;-
                                0590  1204
          70 C8000010 8F    CA  0590  1205         BICL    #PSL$M_CM!PSL$M_TBIT! -
                                0597  1206                 PSL$M_FPD!PSL$M_TP,-(R0); CLEAR CM,T,TP,FPD
          70 000005A4'EF    9E  0597  1207         MOVAB   L^NORMAL,-(R0)          ; PUSH ADDRESS TO RESUME AT IN PREV MODE[14]
                          0C    BA  059E  1208         POPR    #^M<R2,R3>              ; RESTORE WORKING REGISTERS
                       5E 50    D0  05A0  1209         MOVL    R0,SP                   ; POINT TO PC/PSL PAIR
                                02  05A3  1210         REI                             ; RETURN IN PREVIOUS MODE AT 'NORMAL'
```

```
                        05A4  1212 ;+
                        05A4  1213 ; HERE TO BUILD THE CONDITION ARGLIST AND CALL THE HANDLER
                        05A4  1214 ;-
                        05A4  1215
                        05A4  1216 NORMAL:
                  6E    DF  05A4  1217          PUSHAL  (SP)                    ; POINT TO MECHANISM ARRAY
               18 AE    DF  05A6  1218          PUSHAL  24(SP)                  ; POINT TO SIGNAL ARRAY
                  02    DD  05A9  1219          PUSHL   #2                      ; INDICATE ARGLIST LENGTH
                        05AB  1220
                        05AB  1221 .
                        05AB  1222 ; SEARCH FOR CONDITION HANDLER
                        05AB  1223 ;
                        05AB  1224
   000005F6'EF    6E    FA  05AB  1225 120$:   CALLG   (SP),L^SEARCH           ;SEARCH FOR CONDITION HANDLER          [14]
            0B 50    E9  05B2  1226          BLBC    R0,130$                 ;IF LBC TRY LAST CHANCE
                        05B5  1227
                        05B5  1228 ;
                        05B5  1229 ; CALL CONDITION HANDLER
                        05B5  1230 ;
      00000000'9F    16  05B5  1231          JSB     @#SYS$CALL_HANDL        ;CALL HANDLER VIA SYSTEM VECTOR
            ED 50    E9  05BB  1232          BLBC    R0,120$                 ;IF LBS TRY AGAIN
               26    11  05BE  1233          BRB     140$                    ;CONTINUE
                        05C0  1234
                        05C0  1235 ;+
                        05C0  1236 ; HERE WHEN "RESIGNALLED"  (I.E., CONDITION HANDLER DIDN'T WANT IT)
                        05C0  1237 ;-
                        05C0  1238
   FFFFFA39 EF    6E    FA  05C0  1239 130$:   CALLG   (SP), L^COND_HANDLR     ; Call last chance condition handler   [14]
               1C 50    E8  05C7  1240          BLBS    R0,140$                 ; HANDLED, CONTINUE
                        05CA  1241
                        05CA  1242          ERRSUP_S  ,T_YCHFAILURE         ; UNRECOVERABLE SITUATION !
                        05DF  1243          $DS_ABORT                       ; NO OTHER WAY AT THIS POINT
                        05E6  1244 ;+
                        05E6  1245 ; HERE TO CLEAN THE STACK AND RETURN
                        05E6  1246 ;-
                        05E6  1247
                        05E6  1248 140$:
      6E    20 AE    07  C1  05E6  1249          ADDL3   #7,32(SP),(SP)          ; CALCULATE LONGWORD OFFSET TO SAVED PC
               6E    04  C4  05EB  1250          MULL2   #4,(SP)                 ; CALCULATE NUMBER OF BYTES TO REMOVE
            50    18 AE  7D  05EE  1251          MOVQ    24(SP),R0               ; RESTORE R0,R1
               5E    6E  C0  05F2  1252          ADDL2   (SP),SP                 ; REMOVE ARGUMENT LIST FROM STACK
                     02  05F5  1253          REI
```

```
                        05F6  1255          .SBTTL  SEARCH FOR CONDITION HANDLER
                        05F6  1256 ;
                        05F6  1257 ; SEARCH - SEARCH FOR CONDITION HANDLER
                        05F6  1258 ;
                        05F6  1259 ; THIS IS A SPECIAL INTERNAL ROUTINE THAT IS CALLED IN THE INITIAL SEARCH
                        05F6  1260 ; FOR A CONDITION HANDLER AND ON RESIGNAL FROM A PREVIOUSLY SIGNALLED
                        05F6  1261 ; CONDITION.
                        05F6  1262 ;
                        05F6  1263 ; INPUTS:
                        05F6  1264 ;
                        05F6  1265 ;       00(AP) = NUMBER OF CONDITION ARGUMENTS.
                        05F6  1266 ;       04(AP) = ADDRESS OF SIGNAL ARGUMENT LIST.
                        05F6  1267 ;       08(AP) = ADDRESS OF MECHANISM ARGUMENT LIST.
                        05F6  1268 ;       12(AP) = NUMBER OF MECHANISM ARGUMENTS.
                        05F6  1269 ;       16(AP) = FP OF HANDLER ESTABLISHER FRAME.
                        05F6  1270 ;       20(AP) = FRAME DEPTH.
                        05F6  1271 ;       24(AP) = SAVED R0.
                        05F6  1272 ;       28(AP) = SAVED R1.
                        05F6  1273 ;       32(AP) = NUMBER OF SIGNAL ARGUMENTS.
                        05F6  1274 ;       36(AP) = EXCEPTION NAME (INTEGER VALUE).
                        05F6  1275 ;       40(AP) = FIRST EXCEPTION PARAMETER (IF ANY).
                        05F6  1276 ;       44(AP) = SECOND EXCEPTION PARAMETER (IF ANY).
                        05F6  1277 ;              .
                        05F6  1278 ;              .
                        05F6  1279 ;              .
                        05F6  1280 ;       36+N*4(AP) = N'TH EXCEPTION PARAMETER (IF ANY).
                        05F6  1281 ;       36+N*4+4(AP) = EXCEPTION PC.
                        05F6  1282 ;       36+N*4+8(AP) = EXCEPTION PSL.
                        05F6  1283 ;
                        05F6  1284 ; OUTPUTS:
                        05F6  1285 ;
                        05F6  1286 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO LOCATE CONDITION HANDLER.
                        05F6  1287 ;
                        05F6  1288 ;              R0 = SS$_ACCVIO - STACK CANNOT BE READ FROM CURRENT MODE.
                        05F6  1289 ;
                        05F6  1290 ;              R0 = SS$_NOHANDLER - NO CONDITION HANDLER COULD BE FOUND.
                        05F6  1291 ;
                        05F6  1292 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                        05F6  1293 ;
                        05F6  1294 ;              R1 = ADDRESS OF CONDITION HANDLER.
                        05F6  1295 ;
                        05F6  1296
00000000                05F6  1297          HANDLER = 0                     ; CONDITION HANDLER ADDRESS
00000004                05F6  1298          SAVPSW  = 4                     ; SAVED PSW FROM CALL
00000006                05F6  1299          SAVMSK  = 6                     ; REGISTER SAVE MASK
00000008                05F6  1300          SAVAP   = 8                     ; SAVED AP REGISTER IMAGE
0000000C                05F6  1301          SAVFP   = 12                    ; SAVED FP REGISTER IMAGE
00000010                05F6  1302          SAVPC   = 16                    ; SAVED PC REGISTER IMAGE
00000014                05F6  1303          SAVRG   = 20                    ; OTHER SAVED REGISTER IMAGES
```

```
                        05F6  1305 SEARCH:                                ;SEARCH FOR CONDITION HANDLER
                0000    05F6  1306          .WORD   0                     ;ENTRY MASK
    6D  0000069C'EF     DE    05F8  1307          MOVAL   L^CHANDL,(FP)        ;SET ADDRESS OF CONDITION HANDLER       [14]
                        05FF  1308
      50    10 AC       D0    05FF  1309 10$:     MOVL    16(AP),R0            ;GET PREVIOUS FRAME ADDRESS
                        0603  1310
            51          DC    0603  1311 20$:     MOVPSL  R1                   ;READ CURRENT PSL
  51    51    02        18    EF    0605  1312          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1 ;EXTRACT CURRENT MODE
              14 AC     D6    060A  1313          INCL    20(AP)               ;INCREMENT FRAME DEPTH
                        28    13    060D  1314          BEQL    50$                  ;IF EQL FIRST STACK FRAME
                        18    14    060F  1315          BGTR    40$                  ;IF GTR OTHER STACK FRAME
  50    00000007'9F41       7E    0611  1316          MOVAQ   @#CTL$AQ_EXCVEC[R1],R0 ;GET ADDRESS OF EXCEPTION VECTOR QUADWORD
        14 AC   FE 8F       91    0619  1317          CMPB    #-2,20(AP)           ;EXAMINE PRIMARY VECTOR?
                        02    13    061E  1318          BEQL    30$                  ;IF EQL YES
                        80    D5    0620  1319          TSTL    (R0)+                ;ADJUST TO SECONDARY VECTOR
                        0622  1320
            51          60    D0    0622  1321 30$:     MOVL    (R0),R1              ;GET ADDRESS OF CONDITION HANDLER
                        37    12    0625  1322          BNEQ    60$                  ;IF NEQ CONDITION HANDLER FOUND
                        D6    11    0627  1323          BRB     10$                  ;
                        0629  1324
          69 16 AC      E8    0629  1325 40$:     BLBS    22(AP),100$          ;IF LBS SEARCH COUNT OVERFLOW
      50    0C A0       D0    062D  1326          MOVL    SAVFP(R0),R0         ;GET ADDRESS OF PREVIOUS FRAME
                        63    13    0631  1327          BEQL    100$                 ;IF EQL NONE
          :0 AC         50    D0    0633  1328          MOVL    R0,16(AP)            ;SAVE ADDRESS OF ESTABLISHER FRAME
                        0637  1329
00000027'EF41           50    D1    0637  1330 50$:     CMPL    R0,CTL$AL_STACK[R1]  ;FRAME POINTER WITHIN STACK RANGE?
                        55    1A    063F  1331          BGTRU   100$                 ;IF GTRU NO
          00000000'EF   9F    0641  1332          PUSHAB  STACK_BASE           ;SET LOWER LIMIT OF USER STACK
              51        03    D1    0647  1333          CMPL    #PSL$C_USER,R1       ;CURRENT MODE USER?
                        08    13    064A  1334          BEQL    55$                  ;IF EQL YES
  6E    00000023'EF41       D0    064C  1335          MOVL    CTL$AL_STACK-4[R1],(SP) ;SET CURRENT MODE STACK LIMIT
                        0654  1336
            8E      50    D1    0654  1337 55$:     CMPL    R0,(SP)+             ;FRAME POINTER WITHIN STACK RANGE?
                        3D    1F    0657  1338          BLSSU   100$                 ;IF LSSU NO
            51          60    D0    0659  1339          MOVL    (R0),R1              ;GET ADDRESS OF CONDITION HANDLER
                        04    13    065C  1340          BEQL    70$                  ;IF EQL NONE
                        065E  1341
            50      01  C8    065E  1342 60$:     BISL    #1,R0                ;INDICATE SUCCESSFUL COMPLETION
                        04    0661  1343          RET                          ;
                        0662  1344
 10 A0      00000004'8F   D1    0662  1345 70$:     CMPL    #SYS$CALL_HANDL+4,SAVPC(R0) ;CALL FROM CONDITION DISPATCHER?
                        93    12    066A  1346          BNEQ    10$                  ;IF NEQ NO
51  06 A0      0C      00    EF    066C  1347          EXTZV   #0,#12,SAVMSK(R0),R1 ;GET REGISTER SAVE MASK
7E  06 A0      02      0E    EF    0672  1348          EXTZV   #14,#2,SAVMSK(R0),-(SP) ;GET STACK ALIGNMENT BIAS
                50      14    C0    0678  1349          ADDL    #SAVRG,R0            ;ADD OFFSET TO REGISTER SAVE AREA
                50      8E    C0    067B  1350          ADDL    (SP)+,R0             ;ADD STACK ALIGNMENT BIAS
                        067E  1351
              03 51     E9    067E  1352 80$:     BLBC    R1,90$               ;IF LBC CORRESPONDING REGISTER NOT SAVED
                50      04    C0    0681  1353          ADDL    #4,R0                ;ADJUST FOR SAVED REGISTER
                        0684  1354
    51    51    FF 8F   78    0684  1355 90$:     ASHL    #-1,R1,R1            ;ANY MORE REGISTERS SAVED?
                        F3    12    0689  1356          BNEQ    80$                  ;IF NEQ YES
          50    0C A0       D0    068B  1357          MOVL    CHF$L_MCHARGLST+4(R0),R0 ;GET ADDRESS OF MECHANISM ARGUMENTS
          50    04 A0       D0    068F  1358          MOVL    CHF$L_MCH_FRAME(R0),R0 ;GET ADDRESS OF ESTABLISHER FRAME
                  FF6D      31    0693  1359          BRW     20$                  ;
                        0696  1360
        50    08F8 8F     3C    0696  1361 100$:    MOVZWL  #SS$_NOHANDLER,R0    ;SET NO HANDLER FOUND
```

```
                      04  069B  1362          RET                                 ;
                          069C  1363
                          069C  1364 ;
                          069C  1365 ; CONDITION HANDLER FOR SEARCH ROUTINE EXCEPTIONS
                          069C  1366 ;
                          069C  1367
                          069C  1368 CHANDL:                                       ;SEARCH ROUTINE CONDITION HANDLER
                    0000  069C  1369          .WORD   0                           ;ENTRY MASK
          50    04 AC     D0  069E  1370          MOVL    CHF$L_SIGARGLST(AP),R0      ;GET ADDRESS OF SIGNAL ARGUMENTS
    04 A0    0920 8F     B1  06A2  1371          CMPW    #SS$_UNWIND, -              ;UNWINDING?
                          06A8  1372                    CHF$L_SIG_NAME(R0)          ;
                    12    13  06A8  1373          BEQL    10$                         ;IF EQL YES
          51    08 AC     D0  06AA  1374          MOVL    CHF$L_MCHARGLST(AP),R1      ;GET ADDRESS OF MECHANISM ARGUMENTS
       0C A1    04 A0     D0  06AE  1375          MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;SET FINAL STATUS
                   7E     7C  06B3  1376          CLRQ    -(SP)                       ;CLEAR DEPTH AND NEW PC ARGUMENTS
00000000'EF        02     FB  06B5  1377          CALLS   #2,SYS$UNWIND               ;UNWIND TO ESTABLISHER'S CALLER
                          06BC  1378
                      04  06BC  1379 10$:    RET                                 ;
```

```
                                06BD  1381 .SBTTL DSX$SETPRIEXV - ESTABLISH PRIMARY EXCEPTION VECTOR                [19]
                                06BD  1382
                                06BD  1383 ;FUNCTIONAL DESCRIPTION                                                 [19]
                                06BD  1384 ;       This routine services the DS$SETPRIEXV macro call, used by a    [19]
                                06BD  1385 ;       program running under the DS when the progam wishes to establish[19]
                                06BD  1386 ;       a primary exception vector to field exception conditions before [19]
                                06BD  1387 ;       the supervisor's condition handler takes over.                  [19]
                                06BD  1388 ;                                                                       [19]
                                06BD  1389 ;INPUTS                                                                 [19]
                                06BD  1390 ;       4(AP) = Address of the condition handler                        [19]
                                06BD  1391 ;OUTPUTS                                                                [19]
                                06BD  1392 ;       None                                                            [19]
                                06BD  1393 ;                                                                       [19]
                                06BD  1394
                           0004 06BD  1395 .ENTRY  DSX$SETPRIEXV, ^M<R2>           ;                              [19]
                                06BF  1396
                         1C  E1 06BF  1397         BBC     #DSA$V_USER,-          ;IF USER MODE                  [19]
                 12 0000FE00'EF 06C1  1398                 DSA$GL_FLAGS,100$      ;THEN                          [19]
                                06C7  1399         $SETEXV_S       #0,@4(AP)      ;   CALL SYSTEM SERVICE        [19]
                         20  11 06D7  1400         BRB     200$                   ;ELSE                          [19]
      00000007'EF  04 AC  D0 06D9  1401 100$:    MOVL    4(AP),CTL$AQ_EXCVEC    ;   SET KERNEL PRIMARY VECTOR  [19]
      0000000F'EF  04 AC  D0 06E1  1402          MOVL    4(AP),CTL$AQ_EXCVEC+8  ;   SET EXEC   PRIMARY VECTOR  [19]
      00000017'EF  04 AC  D0 06E9  1403          MOVL    4(AP),CTL$AQ_EXCVEC+16 ;   SET SUPER  PRIMARY VECTOR  [19]
      0000001F'EF  04 AC  D0 06F1  1404          MOVL    4(AP),CTL$AQ_EXCVEC+24 ;   SET USER   PRIMARY VECTOR  [19]
                     04  06F9  1405 200$:    RET                            ;RETURN                        [19]
                         06FA  1406
                         06FA  1407         .END
```

```
$$ARGS                          =  0000000B     D
$$N                             =  00000001     D
$$T1                            =  00000000     D
$ER                             =  00000003     D
$MODULE                            00000000 R   D   03
AAT_ARITH                          0000016B RG  D   03
AAT_COMPAT                         000002F0 R   D   03
APC$_ABORT                      =  00000006     D
APC$_CONTINUE                   =  00000009     D
APC$_DUMP                       =  00000003     D
APC$_NOP                        =  00000000     D
APC$_START                      =  00000005     D
APC$_ZERO                       =  00000004     D
APM$_ABTDON                     =  00000006     D
APM$_DEVERR                     =  00000002     D
APM$_DONE                       =  0000000A     D
APM$_EXCEPT                     =  0000000B     D
APM$_HARDERR                    =  00000003     D
APM$_MORE                       =  00000008     D
APM$_NOMES                      =  00000000     D
APM$_PREPERR                    =  0000000C     D
APM$_PRGERR                     =  00000005     D
APM$_SOFTERR                    =  00000004     D
APM$_SPOOL                      =  00000009     D
APM$_SYSERR                     =  00000001     D
APSLMNE                            ********   X   00
BIT...                          =  00000004     D
BPTMAX                             ********   X   00
BUFFER                             FFFFFF00     D
BUFSIZ                          =  00000100     D
CHANDL                             0000069C R   D   04
CHF$L_MCHARGLST                 =  00000008     D
CHF$L_MCH_FRAME                 =  00000004     D
CHF$L_MCH_SAVR0                 =  0000000C     D
CHF$L_SIGARGLST                 =  00000004     D
CHF$L_SIG_NAME                  =  00000004     D
CHR$MCHK                           ********   X   00
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_SIZE                         00000444     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_LAST                         00000024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEST                         00000020     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       0000003C     D
CLI$V_ADAPTER                   =  00000018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BREAK                     =  0000000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
CMK$                            =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
COMMON                             00000126 R   D   04
COND_DEBUG                         ********   X   00
COND_HANDLR                        00000000 RG  D   04
COND_HANDLR_CLI                    0000002F R   D   04
CTL$AL_STACK                       00000027 R   D   03
CTL$AQ_EXCVEC                      00000007 R   D   03
CVTREG$_DATA                    =  00000008     D
CVTREG$_MAXLEN                  =  00000014     D
CVTREG$_MNEADR                  =  0000000C     D
CVTREG$_MSB                     =  00000004     D
CVTREG$_NARGS                   =  0000000B     D
CVTREG$_STRBUF                  =  00000010     D
CVTREG$_V1                      =  00000018     D
```

```
CVTREG$_V2                      =  0000001C     D
CVTREG$_V3                      =  00000020     D
CVTREG$_V4                      =  00000024     D
CVTREG$_V5                      =  00000028     D
CVTREG$_V6                      =  0000002C     D
DEBUG$GB_FLAGS                     ********   X   00
DIR...                          =  FFFFFFFF     D
DS$AA_BPTADDR                      ********   X   00
DS$ABORT                           ********   X   00
DS$CLI                             ********   X   00
DS$CVTREG                          ********   X   00
DS$GA_BREAKVEC                     ********   X   00
DS$GA_TBITVEC                      ********   X   00
DS$GB_TYPECODE                     ********   X   00
DS$GL_FLAGS                        ********   X   00
DS$GW_TTOUT                        ********   X   00
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_PRINTB                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  0000001B     D
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        =  0000000B     D
DS$K_TYPE_FIRST_PASS            =  00000011     D
DS$K_TYPE_GENERAL               =  00000000     D
DS$K_TYPE_GENERAL_ERROR         =  00000003     D
DS$K_TYPE_NC_TESTS              =  00000012     D
DS$K_TYPE_PARAM_ERROR           =  0000001C     D
DS$K_TYPE_PROGRAM_END           =  00000010     D
DS$K_TYPE_PROGRAM_INFO          =  00000017     D
DS$K_TYPE_PROGRAM_START         =  0000000F     D
DS$K_TYPE_QIO_INVADP            =  00000024     D
DS$K_TYPE_QIO_NODRIVER          =  00000022     D
DS$K_TYPE_QIO_WRONGVER          =  00000023     D
DS$K_TYPE_SCRIPT_ECHO           =  00000021     D
DS$K_TYPE_SCRIPT_PNF            =  0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         =  00000020     D
DS$K_TYPE_SCRIPT_SKIP           =  0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        =  00000019     D
DS$K_TYPE_START_ERR             =  00000018     D
DS$K_TYPE_START_LIST            =  00000025     D
DS$K_TYPE_SUMMARY               =  0000000E     D
DS$K_TYPE_USER_PROMPT           =  00000002     D
DS$K_WARNING                    =  00000000     D
DS$M_ABRTFLG                    =  00000040     D
DS$M_BADTIME                    =  00100000     D
DS$M_BATCH                      =  00400000     D
DS$M_BRKCLR                     =  00001000     D
DS$M_BRKPT                      =  00000800     D
DS$M_CHARFLG                    =  00000100     D
DS$M_CMDFLG                     =  00000080     D
DS$M_CTRLC                      =  00000001     D
DS$M_CTRLO                      =  00010000     D
DS$M_DEVFLG                     =  00000200     D
DS$M_DISABLCC                   =  01000000     D
DS$M_DONFLG                     =  00002000     D
DS$M_ERRFLG                     =  00000010     D
DS$M_EXCEPT                     =  00080000     D
DS$M_EXETST                     =  00040000     D
DS$M_HLTFLG                     =  00000008     D
DS$M_LODFLG                     =  00000002     D
DS$M_MEMMGT                     =  00008000     D
DS$M_OUTPUT                     =  00800000     D
DS$M_RUBFLG                     =  00000020     D
DS$M_SCRIPT                     =  00200000     D
DS$M_SETIMR                     =  02000000     D
DS$M_STRFLG                     =  00000004     D
DS$M_SUBT                       =  00004000     D
DS$M_SYSFLG                     =  00000400     D
DS$M_TIMRON                     =  00020000     D
DS$V_ABRTFLG                    =  00000006     D
DS$V_BADTIME                    =  00000014     D
DS$V_BATCH                      =  00000016     D
DS$V_BRKCLR                     =  0000000C     D
DS$V_BRKPT                      =  0000000B     D
DS$V_CHARFLG                    =  00000008     D
DS$V_CMDFLG                     =  00000007     D
DS$V_CTRLC                      =  00000000     D
DS$V_CTRLO                      =  00000010     D
DS$V_DEVFLG                     =  00000009     D
DS$V_DISABLCC                   =  00000018     D
DS$V_DONFLG                     =  0000000D     D
DS$V_ERRFLG                     =  00000004     D
DS$V_EXCEPT                     =  00000013     D
DS$V_EXETST                     =  00000012     D
DS$V_HLTFLG                     =  00000003     D
DS$V_LODFLG                     =  00000001     D
DS$V_MEMMGT                     =  0000000F     D
DS$V_OUTPUT                     =  00000017     D
DS$V_RUBFLG                     =  00000005     D
DS$V_SCRIPT                     =  00000015     D
DS$V_SETIMR                     =  00000019     D
DS$V_STRFLG                     =  00000002     D
DS$V_SUBT                       =  0000000E     D
DS$V_SYSFLG                     =  0000000A     D
```

```
DS$V_TIMRON                     =  00000011     D
DS$_ARITH                       =  00660000     D
DS$_ASBE                        =  00660118     D
DS$_BADLINK                     =  006600F0     D
DS$_BADTYPE                     =  006600E8     D
DS$_BIIC                        =  00660120     D
DS$_CHME                        =  006600A8     D
DS$_CHMK                        =  006600E0     D
DS$_DEVNAME                     =  00660108     D
DS$_ERROR                       =  00660002     D
DS$_FHWE                        =  00660068     D
DS$_FRAGBUF                     =  00660080     D
DS$_GK_EXE_MCHK                 =  00000000     D
DS$_GK_INITSCB                  =  00000002     D
DS$_GK_INT_WORK                 =  00000003     D
DS$_GK_TST$MCHK                 =  00000001     D
DS$_ICBUSY                      =  006600C8     D
DS$_ICERR                       =  006600C0     D
DS$_IHWE                        =  00660060     D
DS$_ILLCHAR                     =  00660018     D
DS$_ILLPAGCNT                   =  00660078     D
DS$_ILLUNIT                     =  00660100     D
DS$_INSFMEM                     =  00660050     D
DS$_IPL2HI                      =  00660088     D
DS$_IVADDR                      =  00660040     D
DS$_IVVECT                      =  00660038     D
DS$_KRNLSTK                     =  00660090     D
DS$_LOGIC                       =  00660070     D
DS$_MCHK                        =  00660088     D
DS$_MMOFF                       =  00660058     D
DS$_NEEDUNIT                    =  006600F8     D
DS$_NODE                        =  00660128     D
DS$_NOPCS                       =  00660110     D
DS$_NORMAL                      =  00660001     D
DS$_NOSUPPORT                   =  006600B1     D
DS$_NOTDON                      =  00660030     D
DS$_NOTIMP                      =  006600B0     D
DS$_NULLSTR                     =  00660010     D
DS$_OVERFLOW                    =  00660008     D
DS$_POWER                       =  00660098     D
DS$_PROGERR                     =  00660020     D
DS$_SEVERE                      =  00660004     D
DS$_TRANSL                      =  006600A0     D
DS$_TRUNCATE                    =  00660028     D
DS$_UNEXPINT                    =  006600D8     D
DS$_VASFULL                     =  00660048     D
DS$_WARNING                     =  00660000     D
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSA$V_USER                      =  0000001C     D
DSR$FAULT_CLEAR                    ********     X   00
DSR$FIND_USER_PC                   ********     X   00
DSX$PRINTB                         ********     X   00
DSX$PRINTI                         ********     X   00
DSX$PRINTSIG                       00000108 RG  D   04
DSX$PRINTSIGI                      0000011A RG  D   04
DSX$PRINTX                         ********     X   00
DSX$SETPRIEXV                      000006BD RG  D   04
DS_ERRSUP                          ********     X   00
ESTKPTR                            ********     X   00
EXE$ACVIOLAT                       00000480 RG  D   04
EXE$BREAK                          000004A8 RG  D   04
EXE$ROPRAND                        00000470 RG  D   04
EXE$TBIT                           000004D4 RG  D   04
EXE_3ARG                           00000523 R   D   04
EXE_4ARG                           0000051F R   D   04
EXE_ACCVIO                         00000480 RG  D   04
EXE_ARITH                          0000049C RG  D   04
EXE_BREAK                          000004A8 RG  D   04
EXE_CHME                           00000508 RG  D   04
EXE_CHMK                           00000500 RG  D   04
EXE_CHMS                           00000510 RG  D   04
EXE_CHMU                           00000518 RG  D   04
EXE_COMPAT                         00000494 RG  D   04
EXE_EXCEPTION                      00000527 R   D   04
EXE_KRNLSTK                        00000448 RG  D   04
EXE_MCHK                           00000428 RG  D   04
EXE_OPCCUS                         00000468 RG  D   04
EXE_OPCDEC                         00000460 RG  D   04
EXE_POWER                          00000454 RG  D   04
EXE_RADRMOD                        00000478 RG  D   04
EXE_ROPRAND                        00000470 RG  D   04
EXE_TBIT                           000004D4 RG  D   04
EXE_TRANSL                         00000488 RG  D   04
EXE_UNEXPINT                       000004F5 RG  D   04
FIND_USERPC                        000003CD RG  D   04
FMT_CHMX_ARG                       000004DD R   D   03
FMT_ERRPC                          0000041F R   D   03
FMT_ERRPSL                         00000473 R   D   03
FMT_EXCEPT                         000003BD R   D   03
FMT_EXCEP_CODE                     000004C6 R   D   03
FMT_PXXX                           00000527 R   D   03
FMT_UMCHK                          00000383 R   D   03
FMT_UNEXPINT                       000003E7 R   D   03
FMT_UNK_ARG                        000004F0 R   D   03
FMT_UNK_CNT                        00000512 R   D   03
FMT_USER_PC_NF                     00000454 R   D   03
FMT_USRPC                          00000438 R   D   03
FMT_VIRT                           00000494 R   D   03
```

```
HANDLER                         = 00000000     D
IO$M_CANCTRLO                     ********   X   00
IO$_WRITEVBLK                     ********   X   00
IS                              = 00000001     D
ISTKPTR                           ********   X   00
LOCAL                             FFFFFF00     D
MMG$PAGEFAULT                     00000488 RG  D   04
MODELST                           ********   X   00
MSG_ACCVIO                        000000B7 R   D   03
MSG_BREAK                         000000F3 R   D   03
MSG_CHME                          0000012F R   D   03
MSG_CHMK                          0000011B R   D   03
MSG_CHMS                          00000143 R   D   03
MSG_CHMU                          00000157 R   D   03
MSG_COMPAT                        00000107 R   D   03
MSG_DECOVF                        0000028C R   D   03
MSG_FLTDIV                        00000264 R   D   03
MSG_FLTDIV_F                      000002C8 R   D   03
MSG_FLTOVF                        00000250 R   D   03
MSG_FLTOVF_F                      000002B4 R   D   03
MSG_FLTUND                        00000278 R   D   03
MSG_FLTUND_F                      000002DC R   D   03
MSG_INTDIV                        0000023C R   D   03
MSG_INTOVF                        00000228 R   D   03
MSG_KRNLSTK                       0000003F R   D   03
MSG_OPCCUS                        0000007B R   D   03
MSG_OPCDEC                        00000067 R   D   03
MSG_POWER                         00000053 R   D   03
MSG_RADRMOD                       000000A3 R   D   03
MSG_ROPRAND                       0000008F R   D   03
MSG_SUBRNG                        000002A0 R   D   03
MSG_TBIT                          000000DF R   D   03
MSG_TRANSL                        000000CB R   D   03
MSG_UMCHK                         00000037 R   D   03
NEW_AP_FP                         00000004 R   D   02
NEW_PC_PSL                        0000000C R   D   02
NEW_SP                            00000000 R   D   02
NORMAL                            000005A4 R   D   04
PCB_BASE                          ********   X   00
PRINT_CHMX_ARG                    00000348 R   D   04
PRINT_PC_PSL                      00000360 R   D   04
PRINT_TYPE_VA                     00000334 R   D   04
PRINT_UNK_EXC                     0000023A R   D   04
PSL$C_USER                      = 00000003     D
PSL$M_CM                        = 80000000     D
PSL$M_FPD                       = 08000000     D
PSL$M_TBIT                      = 00000010     D
PSL$M_TP                        = 40000000     D
PSL$S_CURMOD                    = 00000002     D
PSL$S_PRVMOD                    = 00000002     D
PSL$V_CM                        = 0000001F     D
PSL$V_CURMOD                    = 00000018     D
PSL$V_IS                        = 0000001A     D
PSL$V_PRVMOD                    = 00000016     D
RESIGNAL                          000003C1 R   D   04
SAVABS...                       = FFFFFF00     D
SAVAP                           = 00000008     D
SAVFP                           = 0000000C     D
SAVMSK                          = 00000006     D
SAVPC                           = 00000010     D
SAVPSW                          = 00000004     D
SAVRG                           = 00000014     D
SCB$L_ACCESS                      00000020     D
SCB$L_ARITH                       00000034     D
SCB$L_BREAK                       0000002C     D
SCB$L_CHME                        00000044     D
SCB$L_CHMK                        00000040     D
SCB$L_CHMS                        00000048     D
SCB$L_CHMU                        0000004C     D
SCB$L_COMPAT                      00000030     D
SCB$L_KNLSTK                      00000008     D
SCB$L_MACHCHK                     00000004     D
SCB$L_OPCCUS                      00000014     D
SCB$L_OPCDEC                      00000010     D
SCB$L_POWER                       0000000C     D
SCB$L_RADRMOD                     0000001C     D
SCB$L_ROPRAND                     00000018     D
SCB$L_RXDB                        000000F8     D
SCB$L_SFTLVL1                     00000084     D
SCB$L_SFTLVL10                    000000A8     D
SCB$L_SFTLVL11                    000000AC     D
SCB$L_SFTLVL12                    000000B0     D
SCB$L_SFTLVL13                    000000B4     D
SCB$L_SFTLVL14                    000000B8     D
SCB$L_SFTLVL15                    000000BC     D
SCB$L_SFTLVL2                     00000088     D
SCB$L_SFTLVL3                     0000008C     D
SCB$L_SFTLVL4                     00000090     D
SCB$L_SFTLVL5                     00000094     D
SCB$L_SFTLVL6                     00000098     D
SCB$L_SFTLVL7                     0000009C     D
SCB$L_SFTLVL8                     000000A0     D
SCB$L_SFTLVL9                     000000A4     D
SCB$L_TBIT                        00000028     D
SCB$L_TIMER                       000000C0     D
SCB$L_TRANSL                      00000024     D
SCB$L_TXDB                        000000FC     D
SCB$L_ZERO                        00000000     D
SCB_IMAGE                         ********   X   00
SCRIPT$STOP                       ********   X   00
SEARCH                            000005F6 R   D   04
SIZ...                          = 00000001     D
SS$_ACCVIO                      = 0000000C     D
SS$_BREAK                       = 00000414     D
SS$_CMODSUPR                    = 0000041C     D
SS$_CMODUSER                    = 00000424     D
SS$_COMPAT                      = 0000042C     D
SS$_CONTINUE                    = 00000001     D
SS$_MCHECK                      = 000002BC     D
SS$_NOHANDLER                   = 000008F8     D
SS$_OPCCUS                      = 00000434     D
SS$_OPCDEC                      = 0000043C     D
SS$_RADRMOD                     = 0000044C     D
SS$_RESIGNAL                    = 00000918     D
```

```
SS$_ROPRAND                     = 00000454     D          X_SUBRNG                        00000314 R  D  04
SS$_TBIT                        = 00000464     D          X_TBIT                          000002DC R  D  04
SS$_UNWIND                      = 00000920     D
SSTKPTR                           ********     X    00
STACK_BASE                        ********     X    00
SYS$CALL_HANDL                    ********     X    00
SYS$QIO                           ********   G      04
SYS$SETEXV                        ********   G      04
SYS$UNWIND                        ********     X    00
TST$MCHK                          00000410 RG  D    04
T_ABORT                           0000068A R   D    03
T_ACCVIO                          000005C5 R   D    03
T_BREAK                           000005FA R   D    03
T_CHME                            00000618 R   D    03
T_CHMK                            00000605 R   D    03
T_CHMS                            0000062E R   D    03
T_CHMU                            00000645 R   D    03
T_COMPAT                          000006A6 R   D    03
T_DECOVF                          00000746 R   D    03
T_FAULT                           00000684 R   D    03
T_FLTDIV                          00000713 R   D    03
T_FLTOVF                          00000701 R   D    03
T_FLTUND                          00000733 R   D    03
T_FLT_ABO                         0000069A R   D    03
T_INTDIV                          000006EA R   D    03
T_INTOVF                          000006D9 R   D    03
T_INTRPT                          00000690 R   D    03
T_KRNLSTK                         00000538 R   D    03
T_OPCCUS                          0000057D R   D    03
T_OPCDEC                          0000055D R   D    03
T_POWER                           00000552 R   D    03
T_RADRMOD                         000005AC R   D    03
T_ROPRAND                         0000059B R   D    03
T_STKVIO                          000006B9 R   D    03
T_SUBRNG                          00000757 R   D    03
T_TBIT                            000005F4 R   D    03
T_TRANSL                          000005DE R   D    03
T_TRAP                            0000067F R   D    03
T_XCHFAILURE                      00000656 R   D    03
USTKPTR                           ********       X  00
X_BREAK                           0000027A R   D    04
X_CHMS                            00000282 R   D    04
X_CHMU                            0000028A R   D    04
X_COMPAT                          00000292 R   D    04
X_DECOVF                          0000030C R   D    04
X_FLTDIV                          000002FC R   D    04
X_FLTDIV_F                        00000324 R   D    04
X_FLTOVF                          000002F4 R   D    04
X_FLTOVF_F                        0000031C R   D    04
X_FLTUND                          00000304 R   D    04
X_FLTUND_F                        000003[illegible] R   D    04
X_INTDIV                          000002EC R   D    04
X_INTOVF                          000002E4 R   D    04
X_OPCCUS                          000002BC R   D    04
X_OPCDEC                          000002C4 R   D    04
X_RADRMOD                         000002CC R   D    04
X_ROPRAND                         000002D4 R   D    04
```

Psect synopsis

PSECT name	Allocation		PSECT No.		Attributes									
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC BYTE
ABS	FFFFFF00	(0.)	01	(1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
WORK	00000014	(20.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	NOVEC LONG
DATA	00000767	(1895.)	03	(3.)	NOPIC	USR	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	NOVEC LONG
CODE	000006FA	(1786.)	04	(4.)	NOPIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=0000000B	159 (1)	159 (1)
$$N	=00000001	862 (1)	207 (1) 208 (1) 209 (1) 210 (1) 211 (1) 212 (1) 213 (1) 214 (1) 215 (1) 216 (1) 217 (1) 218 (1) 219 (1) 220 (1) 221 (1) 222 (1) 243 (1) 244 (1) 245 (1) 246 (1) 247 (1) 248 (1) 249 (1) 250 (1) 251 (1) 252 (1) 859 (1) #-862 (1)
$$T1	=00000000	1399 (1)	1399 (1) 159 (1) 358 (1)
$ER	=00000003	1242 (1)	#-1192 (1) #-1242 (1)
$MODULE	00000000-R	189 (1)	1192 (1) 1242 (1)
AAT_ARITH	0000168-R	225 (1)	
AAT_COMPAT	000002F0-R	254 (1)	#-642 (1)
APC$_ABORT	=00000006	168 (1)	
APC$_CONTINUE	=00000009	168 (1)	
APC$_DUMP	=00000003	168 (1)	
APC$_NOP	=00000000	168 (1)	
APC$_START	=00000005	168 (1)	
APC$_ZERO	=00000004	168 (1)	
APM$_ABTDON	=00000006	168 (1)	
APM$_DEVERR	=00000002	168 (1)	
APM$_DONE	=0000000A	168 (1)	
APM$_EXCEPT	=0000000B	168 (1)	#-365 (1)
APM$_HARDERR	=00000003	168 (1)	
APM$_MORE	=00000008	168 (1)	
APM$_NOMES	=00000000	168 (1)	
APM$_PREPERR	=0000000C	168 (1)	
APM$_PRGERR	=00000005	168 (1)	
APM$_SOFTERR	=00000004	168 (1)	
APM$_SPOOL	=00000009	168 (1)	
APM$_SYSERR	=00000001	168 (1)	#-406 (1)
APSLMNE	00000000-XR		147 (1) 798 (1)
BIT...	=00000004	172 (1)	158 (1) 160 (1) 166 (1) 167 (1) 172 (1)
BPTMAX	00000000-XR		#-1082 (1) 144 (1)
BUFFER	FFFFFF00	474 (1)	798 (1) 799 (1)
BUFSIZ	=00000100	174 (1)	474 (1) #-798 (1)
CHANDL	0000690-R	1368 (1)	1307 (1)
CHF$L_MCHARGLST	=00000008		#-1357 (1) #-1374 (1)
CHF$L_MCH_FRAME	=00000004		#-1358 (1)
CHF$L_MCH_SAVR0	=0000000C		#-1375 (1)
CHF$L_SIGARGLST	=00000004		#-1370 (1)
CHF$L_SIG_NAME	=00000004		#-1372 (1) #-1375 (1)
CHR$MCHK	00000000-XR		146 (1) 534 (1)
CLI$K_BUFSIZ	=00000100	165 (1)	165 (1)
CLI$K_SIZE	00000444	165 (1)	
CLI$L_ADDRESS	00000018	165 (1)	
CLI$L_COMMAND	00000004	165 (1)	
CLI$L_DATA	0000001C	165 (1)	
CLI$L_FLAGS	00000000	165 (1)	

```
CLI$L_LAST                      00000024     165     (1)
CLI$L_NEXT                      00000030     165     (1)
CLI$L_PASS                      0000002C     165     (1)
CLI$L_SUBT                      00000028     165     (1)
CLI$L_TEST                      00000020     165     (1)
CLI$Q_BUFQWD                    00000034     165     (1)
CLI$Q_FILE                      00000008     165     (1)
CLI$Q_SECTION                   00000010     165     (1)
CLI$Q_TIME                      0000043C     165     (1)
CLI$T_BUFFER                    0000003C     165     (1)
CLI$V_ADAPTER                  =00000018     165     (1)
CLI$V_ADR                      =0000000B     165     (1)
CLI$V_ASCII                    =00000013     165     (1)
CLI$V_BREAK                    =0000000A     165     (1)
CLI$V_BRIEF                    =00000018     165     (1)
CLI$V_BYTE                     =0000000D     165     (1)
CLI$V_CLEAR                    =00000002     165     (1)
CLI$V_DEC                      =00000010     165     (1)
CLI$V_DEFAULT                  =0000000C     165     (1)
CLI$V_DEPOSIT                  =00000019     165     (1)
CLI$V_EVENT                    =00000008     165     (1)
CLI$V_EXAM                     =00000005     165     (1)
CLI$V_FLAGS                    =00000009     165     (1)
CLI$V_HEX                      =00000012     165     (1)
CLI$V_KERNEL                   =00000017     165     (1)
CLI$V_LOAD                     =00000006     165     (1)
CLI$V_LONG                     =0000000F     165     (1)
CLI$V_NOTNUF                   =00000001     165     (1)
CLI$V_OCT                      =00000011     165     (1)
CLI$V_PREG                     =0000001A     165     (1)
CLI$V_QA                       =00000007     165     (1)
CLI$V_QACKLOO! LOOPS           =0000001C     165     (1)
CLI$V_QAERRORPRINTS            =0000001B     165     (1)
CLI$V_QAMULTIPLEPASS           =0000001F     165     (1)
CLI$V_QASUBTESTLOOPS           =0000001E     165     (1)
CLI$V_QATESTLOOPS              =0000001D     165     (1)
CLI$V_REG                      =00000014     165     (1)
CLI$V_REQUIRED                 =00000000     165     (1)
CLI$V_RUN                      =00000015     165     (1)
CLI$V_SET                      =00000003     165     (1)
CLI$V_SHOW                     =00000004     165     (1)
CLI$V_VALSEC                   =00000016     165     (1)
CLI$V_WORD                     =0000000E     165     (1)
CMK$_                          =00000004     166     (1)
CMK$_ASTEXIT                   =00000000     166     (1)
CMK$_CANTIM                    =0000000B     166     (1)
CMK$_HIBER                     =00000007     166     (1)
CMK$_INITSCB                   =00000008     166     (1)
CMK$_LAST                      =0000000E     166     (1)
CMK$_MMENABLE                  =00000003     166     (1)
CMK$_RCONSOLE                  =00000001     166     (1)
CMK$_REFRESH                   =00000005     166     (1)
CMK$_SCHDWK                    =00000006     166     (1)
CMK$_SETIMR                    =0000000C     166     (1)
CMK$_SETIPL                    =00000009     166     (1)
CMK$_SETPRT                    =0000000D     166     (1)
CMK$_SGIPR                     =0000000A     166     (1)
```

```
CMK$_TCONSOLE                   =00000002     166    (1)
COMMON                           00000126-R   485    (1)    #-479    (1)
COND_DEBUG                       00000000-XR                   146    (1)      192    (1)      193    (1)      194    (1)
                                                               195    (1)
COND_HANDLR                      00000000-R   355    (1)      1239    (1)
COND_HANDLR_CLI                  0000002F-R   363    (1)    #-360    (1)
CTL$AL_STACK                     00000027-R   197    (1)    #-1330    (1)    #-1335    (1)
CTL$AQ_EXCVEC                    00000007-R   191    (1)      1316    (1)    #-1401    (1)    #-1402    (1)    #-1403    (1)
                                                            #-1404    (1)
CVTREG$_DATA                    =00000008     159    (1)
CVTREG$_MAXLEN                  =00000014     159    (1)
CVTREG$_MNEADR                  =0000000C     159    (1)
CVTREG$_MSB                     =00000004     159    (1)
CVTREG$_NARGS                   =0000000B     159    (1)
CVTREG$_STRBUF                  =00000010     159    (1)
CVTREG$_V1                      =00000018     159    (1)
CVTREG$_V2                      =0000001C     159    (1)
CVTREG$_V3                      =00000020     159    (1)
CVTREG$_V4                      =00000024     159    (1)
CVTREG$_V5                      =00000028     159    (1)
CVTREG$_V6                      =0000002C     159    (1)
DEBUG$GB_FLAGS                   00000000-XR                 #-1104    (1)      145    (1)
DIR...                          =FFFFFFFF     474    (1)      474    (1)
DS$AA_BPTADDR                    00000000-XR                 #-1083    (1)      145    (1)
DS$ABORT                         00000000-XR                   1193    (1)      1243    (1)      144    (1)
DS$CLI                           00000000-XR                    146    (1)      409    (1)
DS$CVTREG                        00000000-XR                    147    (1)      798    (1)
DS$GA_BREAKVEC                   00000000-XR                 #-1078    (1)      145    (1)
DS$GA_TBITVEC                    00000000-XR                 #-1102    (1)      145    (1)
DS$GB_TYPECODE                   00000000-XR                    151    (1)    #-369    (1)    #-371    (1)    #-487    (1)
                                                            #-588    (1)    #-594    (1)    #-771    (1)    #-781    (1)
                                                            #-790    (1)    #-804    (1)    #-809    (1)
DS$GL_FLAGS                      00000000-XR                    148    (1)      403    (1)      404    (1)
DS$GW_TTOUT                      00000000-XR                    151    (1)    #-358    (1)
DS$K_ERROR                      =00000002     160    (1)
DS$K_NORMAL                     =00000001     160    (1)
DS$K_PRINTB                     =00000002     158    (1)
DS$K_PRINTF                     =00000001     158    (1)
DS$K_PRINTI                     =00000000     158    (1)
DS$K_PRINTX                     =00000003     158    (1)
DS$K_SEVERE                     =00000004     160    (1)
DS$K_SUBSYS                     =00000066     160    (1)      160    (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     158    (1)
DS$K_TYPE_ABORT_TEST            =00000013     158    (1)
DS$K_TYPE_COMMAND_ERR           =00000015     158    (1)
DS$K_TYPE_COMMAND_OUT           =00000016     158    (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     158    (1)
DS$K_TYPE_DS_PROMPT             =00000001     158    (1)
DS$K_TYPE_DS_START              =0000001D     158    (1)
DS$K_TYPE_ERRDEV                =00000008     158    (1)
DS$K_TYPE_ERRHARD               =00000006     158    (1)
DS$K_TYPE_ERROR_BODY            =00000009     158    (1)
DS$K_TYPE_ERROR_END             =0000000A     158    (1)
DS$K_TYPE_ERRPREP               =0000001B     158    (1)
DS$K_TYPE_ERRSOFT               =00000007     158    (1)
DS$K_TYPE_ERRSUP                =00000004     153    (1)
DS$K_TYPE_ERRSYS                =00000005     158    (1)
```

```
DS$K_TYPE_ERR_HALT              =0000000D      158     (1)
DS$K_TYPE_EXCEPTION             =0000000C      158     (1)    #-368     (1)    #-593     (1)    #-770     (1)    #-780     (1)
                                                              #-789     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      158     (1)    #-486     (1)    #-587     (1)
DS$K_TYPE_FIRST_PASS            =00000011      158     (1)
DS$K_TYPE_GENERAL               =00000000      158     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      158     (1)
DS$K_TYPE_NO_TESTS              =00000012      158     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      158     (1)
DS$K_TYPE_PROGRAM_END           =00000010      158     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      158     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      158     (1)
DS$K_TYPE_QIO_INVADP            =00000024      158     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      158     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      158     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      158     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      158     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      158     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      158     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      158     (1)
DS$K_TYPE_START_ERR             =00000018      158     (1)
DS$K_TYPE_START_LIST            =0000002F      158     (1)
DS$K_TYPE_SUMMARY               =0000000E      158     (1)
DS$K_TYPE_USER_PROMPT           =00000002      158     (1)
DS$K_WARNING                    =00000000      160     (1)
DS$M_ABRTFLG                    =00000040      167     (1)
DS$M_BADTIME                    =00100000      167     (1)
DS$M_BATCH                      =00400000      167     (1)
DS$M_BRKCLR                     =00001000      167     (1)
DS$M_BRKPT                      =00000800      167     (1)
DS$M_CHARFLG                    =00000100      167     (1)
DS$M_CMDFLG                     =00000080      167     (1)
DS$M_CTRLC                      =00000001      167     (1)
DS$M_CTRLO                      =00010000      167     (1)
DS$M_DEVFLG                     =00000200      167     (1)
DS$M_DISABLCC                   =01000000      167     (1)
DS$M_DONFLG                     =00002000      167     (1)
DS$M_ERRFLG                     =00000010      167     (1)
DS$M_EXCEPT                     =00080000      167     (1)
DS$M_EXETST                     =00040000      167     (1)
DS$M_HLTFLG                     =00000008      167     (1)
DS$M_LODFLG                     =00000002      167     (1)
DS$M_MEMMGT                     =00008000      167     (1)
DS$M_OUTPUT                     =00800000      167     (1)
DS$M_RUBFLG                     =00000020      167     (1)
DS$M_SCRIPT                     =00200000      167     (1)
DS$M_SETIMR                     =02000000      167     (1)
DS$M_STRFLG                     =00000004      167     (1)
DS$M_SUBT                       =00004000      167     (1)
DS$M_SYSFLG                     =00000400      167     (1)
DS$M_TIMRON                     =00020000      167     (1)
DS$V_ABRTFLG                    =00000006      167     (1)
DS$V_BADTIME                    =00000014      167     (1)
DS$V_BATCH                      =00000016      167     (1)
DS$V_BRKCLR                     =0000000C      167     (1)
DS$V_BRKPT                      =00000008      167     (1)
DS$V_CHARFLG                    =00000008      167     (1)
```

```
DS$V_CMDFLG                     =00000007     167     (1)     #-403     (1)
DS$V_CTRLC                      =00000000     167     (1)
DS$V_CTRLO                      =00000010     167     (1)
DS$V_DEVFLG                     =00000009     167     (1)
DS$V_DISABLCC                   =00000018     167     (1)
DS$V_DONFLG                     =0000000D     167     (1)
DS$V_ERRFLG                     =00000004     167     (1)
DS$V_EXCEPT                     =00000013     167     (1)     #-404     (1)
DS$V_EXETST                     =00000012     167     (1)
DS$V_HLTFLG                     =00000003     167     (1)
DS$V_LODFLG                     =00000001     167     (1)
DS$V_MEMMGT                     =0000000F     167     (1)
DS$V_OUTPUT                     =00000017     167     (1)
DS$V_RUBFLG                     =00000005     167     (1)
DS$V_SCRIPT                     =00000C15     167     (1)
DS$V_SETIMR                     =00000019     167     (1)
DS$V_STRFLG                     =00000002     167     (1)
DS$V_SUBT                       =0000000E     167     (1)
DS$V_SYSFLG                     =0000000A     167     (1)
DS$V_TIMRON                     =00000011     167     (1)
DS$_ARITH                       =006600D0     160     (1)     #-1067    (1)     #-564     (1)
DS$_ASBE                        =00660118     160     (1)
DS$_BADLINK                     =006600F0     160     (1)
DS$_BADTYPE                     =006600E8     160     (1)
DS$_BIIC                        =00660120     160     (1)
DS$_CHME                        =006600A8     160     (1)     #-1134    (1)     #-547     (1)
DS$_CHMK                        =006600E0     160     (1)     #-1126    (1)     #-552     (1)
DS$_DEVNAME                     =00660108     160     (1)
DS$_ERROR                       =00660002     160     (1)
DS$_FHWE                        =00660068     160     (1)
DS$_FRAGBUF                     =00660080     160     (1)
DS$_GK_EXE_MCHK                 =00000000     172     (1)     #-969     (1)
DS$_GK_INITSCB                  =00000002     172     (1)
DS$_GK_INT_WORK                 =00000003     172     (1)
DS$_GK_TST$MCHK                 =00000001     172     (1)     #-913     (1)
DS$_ICBUSY                      =006600C8     160     (1)
DS$_ICERR                       =006600C0     160     (1)
DS$_IHWE                        =00660060     160     (1)
DS$_ILLCHAR                     =00660018     160     (1)
DS$_ILLPAGCNT                   =00660078     160     (1)
DS$_ILLUNIT                     =00660100     160     (1)
DS$_INSFMEM                     =00660050     160     (1)
DS$_IPL2HI                      =006600B8     160     (1)
DS$_IVADDR                      =00660040     160     (1)
DS$_IVVECT                      =00660038     160     (1)
DS$_KRNLSTK                     =00660090     160     (1)     #-537     (1)     #-980     (1)
DS$_LOGIC                       =00660070     160     (1)
DS$_MCHK                        =00660088     160     (1)     #-530     (1)     #-965     (1)
DS$_MMOFF                       =00660058     160     (1)
DS$_NEEDUNIT                    =006600F8     160     (1)
DS$_NODE                        =00660128     160     (1)
DS$_NOPCS                       =00660110     160     (1)
DS$_NORMAL                      =00660001     160     (1)
DS$_NOSUPPORT                   =006600B1     160     (1)
DS$_NOTDON                      =00660030     160     (1)
DS$_NOTIMP                      =006600B0     160     (1)       160     (1)
DS$_NULLSTR                     =00660010     160     (1)
```

```
DS$_OVERFLOW                    =00660008      160    (1)
DS$_POWER                       =00660098      160    (1)    #-969    (1)
DS$_PROGERR                     =00660020      160    (1)
DS$_SEVERE                      =00660004      160    (1)
DS$_TRANSL                      =006600A0      160    (1)    #-1048   (1)    #-542    (1)
DS$_TRUNCATE                    =00660028      160    (1)
DS$_UNEXPINT                    =006600D8      160    (1)    #-1117   (1)    #-557    (1)
DS$_VASFULL                     =00660048      160    (1)
DS$_WARNING                     =00660000      160    (1)      160    (1)
DSA$GL_FLAGS                     0000FE00                      1398   (1)      364    (1)      384    (1)
DSA$GL_MSGTYP                    0000FE40                    #-366    (1)    #-407    (1)
DSA$V_USER                      =0000001C                    #-1397   (1)    #-364    (1)    #-384    (1)
DSR$FAULT_CLEAR                  00000000-XR                   149    (1)      914    (1)      970    (1)
DSR$FIND_USER_PC                 00000000-XR                   152    (1)      855    (1)
DSX$PRINTB                       00000000-XR                   148    (1)      477    (1)
DSX$PRINTI                       00000000-XR                   144    (1)      482    (1)      859    (1)      862    (1)
DSX$PRINTSIG                     00000108-R     476    (1)
DSX$PRINTSIGI                    0000011A-R     481    (1)    359    (1)
DSX$PRINTX                       00000000-XR                   149    (1)      478    (1)
DSX$SETPRIEXV                    000006BD-R     1395   (1)
DS_ERRSUP                        00000000-XR                  1192    (1)     1242    (1)      144    (1)
ESTKPTR                          00000000-XR                   150    (1)      198    (1)
EXE$ACVIOLAT                     00000480-R     1036   (1)
EXE$BREAK                        000004A8-R     1077   (1)
EXE$ROPRAND                      00000470-R     1017   (1)
EXE$TBIT                         000004D4-R     1101   (1)
EXE_3ARG                         00000523-R     1159   (1)    #-1000   (1)    #-1009   (1)    #-1019   (1)    #-1028   (1)
                                                             #-1091   (1)    #-1111   (1)    #-981    (1)    #-991    (1)
EXE_4ARG                         0000051F-R     1155   (1)    #-1059   (1)    #-1068   (1)    #-1118   (1)    #-1127   (1)
                                                             #-1135   (1)    #-1144   (1)    #-1153   (1)
EXE_ACCVIO                       00000480-R     1035   (1)
EXE_ARITH                        0000049C-R     1066   (1)
EXE_BREAK                        000004A8-R     1076   (1)
EXE_CHME                         00000508-R     1133   (1)
EXE_CHMK                         00000500-R     1125   (1)
EXE_CHMS                         00000510-R     1142   (1)
EXE_CHMU                         00000518-R     1151   (1)
EXE_COMPAT                       00000494-R     1057   (1)    #-1109   (1)
EXE_EXCEPTION                    00000527-R     1169   (1)    #-1039   (1)    #-1050   (1)    #-1157   (1)    #-1161   (1)
                                                             #-972    (1)
EXE_KRNLSTK                      00000448-R     979    (1)
EXE_MCHK                         00000428-R     964    (1)
EXE_OPCCUS                       00000468-R     1007   (1)
EXE_OPCDEC                       00000460-R     998    (1)
EXE_POWER                        00000454-R     988    (1)
EXE_RADRMOD                      00000478-R     1026   (1)
EXE_ROPRAND                      00000470-R     1016   (1)
EXE_TBIT                         000004D4-R     1100   (1)
EXE_TRANSL                       00000488-R     1046   (1)
EXE_UNEXPINT                     000004F5-R     1116   (1)
FIND_USERPC                      000003CD-R     848    (1)    370    (1)
FMT_CHMX_ARG                     000004DD-R     281    (1)    784    (1)
FMT_ERRPC                        0000041F-R     275    (1)    794    (1)
FMT_ERRPSL                       00000473-R     278    (1)    802    (1)
FMT_EXCEPT                       000003BD-R     273    (1)    208    (1)      209    (1)      210    (1)      211    (1)
                                                             212    (1)      213    (1)      214    (1)      215    (1)
                                                             216    (1)      217    (1)      218    (1)      219    (1)
```

```
                                                        220  (1)     221  (1)     222  (1)     243  (1)
                                                        244  (1)     245  (1)     246  (1)     247  (1)
                                                        248  (1)     249  (1)     250  (1)     251  (1)
                                                        252  (1)
FMT_EXCEP_CODE                  000004C6-R  280  (1)    645  (1)
FMT_PXXX                        00000527-R  284  (1)    605  (1)
FMT_UMCHK                       00000383-R  271  (1)    207  (1)
FMT_UNEXPINT                    000003E7-R  274  (1)    560  (1)
FMT_UNK_ARG                     000004F0-R  282  (1)    590  (1)
FMT_UNK_CNT                     00000512-R  283  (1)    596  (1)
FMT_USER_PC_NF                  00000454-R  277  (1)    862  (1)
FMT_USRPC                       00000438-R  276  (1)    859  (1)
FMT_VIRT                        00000494-R  279  (1)    775  (1)
HANDLER                        =00000000    1297 (1)
IO$M_CANCTRLO                   00000000-XR             150  (1)   #-358   (1)
IO$_WRITEVBLK                   00000000-XR             151  (1)   #-358   (1)
IS                             =00000001    173  (1)
ISTKPTR                         00000000-XR             150  (1)     198  (1)
LOCAL                           FFFFFF00    474  (1)    489  (1)
MMG$PAGEFAULT                   00000488-R  1047 (1)
MODELST                         00000000-XR             147  (1)     798  (1)
MSG_ACCVIO                      000000B7-R  214  (1)    522  (1)
MSG_BREAK                       000000F3-R  217  (1)    616  (1)
MSG_CHME                        0000012F-R  220  (1)    549  (1)
MSG_CHMK                        0000011B-R  219  (1)    554  (1)
MSG_CHMS                        00000143-R  221  (1)    624  (1)
MSG_CHMU                        00000157-R  222  (1)    632  (1)
MSG_COMPAT                      00000107-R  218  (1)    640  (1)
MSG_DECOVF                      0000028C-R  248  (1)    733  (1)
MSG_FLTDIV                      00000264-R  246  (1)    717  (1)
MSG_FLTDIV_F                    000002C8-R  251  (1)    757  (1)
MSG_FLTOVF                      00000250-R  245  (1)    709  (1)
MSG_FLTOVF_F                    000002B4-R  250  (1)    749  (1)
MSG_FLTUND                      00000278-R  247  (1)    725  (1)
MSG_FLTUND_F                    000002DC-R  252  (1)    766  (1)
MSG_INTDIV                      0000023C-R  244  (1)    701  (1)
MSG_INTOVF                      00000228-R  243  (1)    694  (1)
MSG_KRNLSTK                     0000003F-R  208  (1)    539  (1)
MSG_OPCCUS                      0000007B-R  211  (1)    654  (1)
MSG_OPCDEC                      00000067-R  210  (1)    662  (1)
MSG_POWER                       00000053-R  209  (1)
MSG_RADRMOD                     000000A3-R  213  (1)    670  (1)
MSG_ROPRAND                     0000008F-R  212  (1)    678  (1)
MSG_SUBRNG                      000002A0-R  249  (1)    741  (1)
MSG_TBIT                        000000DF-R  216  (1)    686  (1)
MSG_TRANSL                      000000CB-R  215  (1)    544  (1)
MSG_UMCHK                       00000037-R  207  (1)    527  (1)
NEW_AP_FP                       00000004-R  182  (1)  #-415  (1)   #-424   (1)
NEW_PC_PSL                      0000000C-R  184  (1)  #-417  (1)   #-426   (1)
NEW_SP                          00000000-R  180  (1)  #-416  (1)   #-425   (1)
NORMAL                          000005A4-R  1216 (1)  #-1176 (1)   #-1179  (1)    1207  (1)
PCB_BASE                        00000000-XR           #-1185 (1)   #-1195  (1)     148  (1)  #-392   (1)
PRINT_CHMX_ARG                  0000034B-R  779  (1)  #-555  (1)
PRINT_PC_PSL                    00000360-R  788  (1)    492  (1)     566  (1)  #-609   (1)  #-786   (1)
PRINT_TYPE_VA                   00000334-R  769  (1)  #-523  (1)   #-545   (1)
PRINT_UNK_EXC                   0000023A-R  586  (1)  #-579  (1)
PSL$C_USER                     =00000003            #-1333 (1)
```

```
PSL$M_CM                        =80000000                       #-1205    (1)
PSL$M_FPD                       =08000000                       #-1206    (1)
PSL$M_TBIT                      =00000010                       #-1205    (1)
PSL$M_TP                        =40000000                       #-1206    (1)
PSL$S_CURMOD                    =00000002                       #-1312    (1)   #-386    (1)   #-389    (1)
PSL$S_PRVMOD                    =00000002                       #-1178    (1)
PSL$V_CM                        =0000001F                       #-1108    (1)
PSL$V_CURMOD                    =00000018                       #-1312    (1)   #-386    (1)   #-389    (1)
PSL$V_IS                        =0000001A       166     (1)     #-1176    (1)   #-383    (1)
PSL$V_PRVMOD                    =00000016                       #-1177    (1)
RESIGNAL                         000003C1-R     807     (1)       518     (1)
SAVABS...                       =FFFFFF00       474     (1)
SAVAP                           =00000008       1300    (1)
SAVFP                           =0000000C       1301    (1)     #-1326    (1)
SAVMSK                          =00000006       1299    (1)       1347    (1)     1348    (1)
SAVPC                           =00000010       1302    (1)     #-1345    (1)
SAVPSW                          =00000004       1298    (1)
SAVRG                           =00000014       1303    (1)     #-1349    (1)
SCB_IMAGE                        00000000-XR                      152     (1)
SCRIPT$STOP                      00000000-XR                      149     (1)   #-408    (1)
SEARCH                           000005F6-R     1305    (1)      1225     (1)
SIZ...                          =00000001       167     (1)       167     (1)
SS$_ACCVIO                      =0000000C                       #-1037    (1)   #-520    (1)
SS$_BREAK                       =00000414                       #-1090    (1)   #-518    (1)
SS$_CMODSUPR                    =0000041C                       #-1143    (1)
SS$_CMODUSER                    =00000424                       #-1152    (1)
SS$_COMPAT                      =0000042C                       #-1058    (1)
SS$_CONTINUE                    =00000001                       #-421     (1)
SS$_MCHECK                      =000002BC                       #-525     (1)
SS$_NOHANDLER                   =000008F8                       #-1361    (1)
SS$_OPCCUS                      =00000434                       #-1008    (1)
SS$_OPCDEC                      =0000043C                       #-999     (1)
SS$_RADRMOD                     =0000044C                       #-1027    (1)
SS$_RESIGNAL                    =00000918                       #-808     (1)
SS$_ROPRAND                     =00000454                       #-1018    (1)
SS$_TBIT                        =00000464                       #-1110    (1)
SS$_UNWIND                      =00000920                       #-1371    (1)
SSTRPTR                          00000000-XR                      150     (1)     198     (1)
STACK_BASE                       00000000-XR                     1332     (1)     148     (1)
SYS$CALL_HANDL                   00000000-XR                     1231     (1)   #-1345    (1)     149     (1)
SYS$QIC                          00000000-XR                      358     (1)
SYS$SETEXV                       00000000-XR                     1399     (1)
SYS$UNWIND                       00000000-XR                     1377     (1)     146     (1)
TST$MCHK                         00000410-R     911     (1)
T_ABORT                          0000068A-R     303     (1)      208     (1)
T_ACCVIO                         000005C5-R     292     (1)      214     (1)
T_BREAK                          000005FA-R     295     (1)      217     (1)
T_CHME                           00000618-R     297     (1)      220     (1)
T_CHMK                           00000605-R     296     (1)      219     (1)
T_CHMS                           0000062E-R     298     (1)      221     (1)
T_CHMU                           00000645-R     299     (1)      222     (1)
T_COMPAT                         000006A6-R     306     (1)      218     (1)     644     (1)
T_DECOVF                         00000746-R     313     (1)      248     (1)
T_FAULT                          00000684-R     302     (1)      210     (1)     211     (1)     213     (1)     214     (1)
                                                                 215     (1)     217     (1)     250     (1)     251     (1)
                                                                 252     (1)
T_FLTDIV                         00000713-R     311     (1)      246     (1)     251     (1)
```

```
T_FLTOVF                00000701-R  310 (1)  245 (1)  250 (1)
T_FLTUND                00000733-R  312 (1)  247 (1)  252 (1)
T_FLT_ABO               0000069A-R  305 (1)  212 (1)  218 (1)  643 (1)
T_INTDIV                000006EA-R  309 (1)  244 (1)
T_INTOVF                000006D9-R  308 (1)  243 (1)
T_INTRPT                00000690-R  304 (1)  209 (1)
T_KRNLSTK               0000053B-R  286 (1)  208 (1)
T_OPCCUS                0000057D-R  289 (1)  211 (1)
T_OPCDEC                0000055D-R  288 (1)  210 (1)
T_POWER                 00000552-R  287 (1)  209 (1)
T_RADRMOD               000005AC-R  291 (1)  213 (1)
T_ROPRAND               0000059B-R  290 (1)  212 (1)
T_STKVIO                000006B9-R  307 (1)  1192 (1)
T_SUBRNG                00000757-R  314 (1)  249 (1)
T_TBIT                  000005F4-R  294 (1)  216 (1)
T_TRANSL                000005DE-R  293 (1)  215 (1)
T_TRAP                  0000067F-R  301 (1)  216 (1)  219 (1)  220 (1)  221 (1)
                                             222 (1)  243 (1)  244 (1)  245 (1)
                                             246 (1)  247 (1)  248 (1)  249 (1)
T_XCHFAILURE            00000656-R  300 (1)  1242 (1)
USTKPTR                 00000000-XR          147 (1)  198 (1)
X_BREAK                 0000027A-R  615 (1)  518 (1)
X_CHMS                  00000282-R  623 (1)  518 (1)
X_CHMU                  0000028A-R  631 (1)  518 (1)
X_COMPAT                00000292-R  639 (1)  518 (1)
X_DECOVF                0000030C-R  732 (1)  518 (1)  578 (1)
X_FLTDIV                000002FC-R  716 (1)  518 (1)  578 (1)
X_FLTDIV_F              00000324-R  756 (1)  518 (1)  578 (1)
X_FLTOVF                000002F4-R  708 (1)  518 (1)  578 (1)
X_FLTOVF_F              0000031C-R  748 (1)  518 (1)  578 (1)
X_FLTUND                00000304-R  724 (1)  518 (1)  578 (1)
X_FLTUND_F              0000032C-R  765 (1)  518 (1)  578 (1)
X_INTDIV                000002EC-R  700 (1)  518 (1)  578 (1)
X_INTOVF                000002E4-R  693 (1)  518 (1)  578 (1)
X_OPCCUS                000002BC-R  653 (1)  518 (1)
X_OPCDEC                000002C4-R  661 (1)  518 (1)
X_RADRMOD               000002CC-R  669 (1)  518 (1)
X_ROPRAND               000002D4-R  677 (1)  518 (1)
X_SUBRNG                00000314-R  740 (1)  518 (1)  578 (1)
X_TBIT                  000002DC-R  685 (1)  518 (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CHFDEF	1	170 (1)	170 (1)
$D1_ABPROGRAM	1	1193 (1)	1193 (1) 1243 (1)
$D1_PRINT_L	2	207 (1)	207 (1) 208 (1) 209 (1) 210 (1) 211 (1)
			212 (1) 213 (1) 214 (1) 215 (1) 216 (1)
			217 (1) 218 (1) 219 (1) 220 (1) 221 (1)
			222 (1) 243 (1) 244 (1) 245 (1) 246 (1)
			247 (1) 248 (1) 249 (1) 250 (1) 251 (1)
			252 (1)
$D1_PRINT_S	1	859 (1)	859 (1) 862 (1)
$DEF	1	172 (1)	
$DEFINI	1	161 (1)	161 (1) 162 (1) 163 (1) 164 (1) 169 (1)
			170 (1)
$DS_ABORT	1	1193 (1)	1193 (1) 1243 (1)
$DS_CVTREG_DEF	1	159 (1)	159 (1)
$DS_CVTREG_S	1	796 (1)	796 (1)
$DS_DSADEF	5	169 (1)	169 (1)
$DS_DSDEF	2	160 (1)	160 (1)
$DS_SCBDEF	3	163 (1)	163 (1)
$DS_TYPEDEF	4	158 (1)	158 (1)
$EQU	1	172 (1)	158 (1) 160 (1) 166 (1) 172 (1)
$EQULS1	1	172 (1)	158 (1) 160 (1) 166 (1) 172 (1)
$EQULST	1	158 (1)	158 (1) 160 (1) 166 (1) 172 (1)
$GBLINI	2	158 (1)	158 (1) 160 (1) 166 (1) 167 (1) 172 (1)
$OFFDEF	1	159 (1)	159 (1)
$OFFSET	2	472 (1)	472 (1)
$OFFST1	1	474 (1)	474 (1)
$PRDEF	4	161 (1)	161 (1)
$PRINTI_L	1	207 (1)	207 (1) 208 (1) 209 (1) 210 (1) 211 (1)
			212 (1) 213 (1) 214 (1) 215 (1) 216 (1)
			217 (1) 218 (1) 219 (1) 220 (1) 221 (1)
			222 (1) 243 (1) 244 (1) 245 (1) 246 (1)
			247 (1) 248 (1) 249 (1) 250 (1) 251 (1)
			252 (1)
$PRINTI_S	1	859 (1)	859 (1) 862 (1)
$PSLDEF	2	162 (1)	162 (1)
$PUSHADR	1	358 (1)	1192 (1) 1242 (1) 1399 (1) 358 (1) 798 (1)
$PUSHTWO	1	358 (1)	358 (1)
$QIOPUSH	1	358 (1)	1399 (1) 358 (1)
$QIO_S	1	357 (1)	357 (1)
$SETEXV_S	1	1399 (1)	1399 (1)
$SSDEF	1	164 (1)	164 (1) 171 (1)
$VIELD	1	167 (1)	167 (1)
$VIELD1	1	172 (1)	167 (1)
APTDEF	1	168 (1)	168 (1)
BR_IF_USER	1	364 (1)	364 (1) 384 (1)
CASE	1	494 (1)	494 (1) 567 (1)
CLIDEF	3	165 (1)	165 (1)
CMKDEF	1	166 (1)	166 (1)
DSFDEF	3	167 (1)	167 (1)
ERRSUP_S	1	1192 (1)	1192 (1) 1242 (1)

FLTCLR_SEL	1	172	(1)	172	(1)
MODNAM	1	189	(1)	189	(1)
SET_CMDFLG	1	403	(1)	403	(1)
SET_EXCEPT	1	404	(1)	404	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.10	00:00:00.31
Command processing	138	00:00:00.73	00:00:02.03
Pass 1	1314	00:00:22.91	00:00:29.88
Symbol table sort	32	00:00:01.75	00:00:01.93
Pass 2	733	00:00:05.61	00:00:09.36
Symbol table output	66	00:00:00.43	00:00:00.67
Psect synopsis output	8	00:00:00.03	00:00:00.06
Cross-reference output	177	00:00:02.12	00:00:02.76
Assembler run totals	2508	00:00:33.69	00:00:47.00

The working set limit was 1000 pages.
158678 bytes (310 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1070 non-local and 63 local symbols.
1407 source lines were read in Pass 1, producing 0 object records in Pass 2.
115 pages of virtual memory were used to define 42 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	10
DRB1:[DS.WORK]DS.MLB;218	12
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	17
TOTALS (all libraries)	40

1187 GETS were required to define 40 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) EXCEPT/UPDA=(EXCEPT.UPD,EXCEPT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 67.2 Libraries, Equated Symbols
(1) 172.2 FAO - Main program
(1) 381 GETCHAR - Routine to get next char from input string
(1) 429 GETCOUNT - Routine to get repeat-count or field-width
(1) 500 CVTASC - Insert ASCII string
(1) 639 CVTNUM - Convert numeric parameter to ASCII
(1) 851 QUICKSERVE - Small service routines
(1) 989 PERCENT - Time directives and plural 'S'
(1) 1083.4 HANDLER - Condition handler

```
                        0000     .1          .TITLE  FAO     *** - FORMATTED ASCII OUTPUT
                        0000     .2          .IDENT  '03-02'
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
                        0000      8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                        0000      9 ;*  ALL RIGHTS RESERVED.                                                     *
                        0000     10 ;*                                                                           *
                        0000     11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                        0000     12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
                        0000     13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
                        0000     14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000     15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
                        0000     16 ;*  TRANSFERRED.                                                             *
                        0000     17 ;*                                                                           *
                        0000     18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
                        0000     19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
                        0000     20 ;*  CORPORATION.                                                             *
                        0000     21 ;*                                                                           *
                        0000     22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
                        0000     23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     24 ;*                                                                           *
                        0000     25 ;*                                                                           *
                        0000     26 ;*****************************************************************************
                        0000     27
                        0000     28 ;++
                        0000     29 ; FACILITY: SYSTEM SERVICE
                        0000     30 ;
                        0000     31 ; ABSTRACT:
                        0000     32 ;
                        0000     33 ;       This module provides general formatting services.  It converts
                        0000     34 ;       binary values to octal, hexadecimal, and decimal ASCII
                        0000     35 ;       representations, and also inserts ASCII strings and converts
                        0000     36 ;       date and time to ASCII.
                        0000     37 ;
                        0000     38 ; ENVIRONMENT:
                        0000     39 ;
                        0000     .1 ;       VAX Diagnostic Supervisor.
                        0000     .2 ;
                        0000     .3 ; AUTHOR:
                        0000     .4 ;
                        0000     .5 ;       Henry M. Levy
                        0000     .6 ;
                        0000     .7 ; CREATION DATE:
                        0000     .8 ;
                        0000     .9 ;       29-JAN-1977
-3                      0000     43 ;
                        0000     44 ; MODIFIED BY:
                        0000     45 ;
                        0000     .1 ;       03-02   Jack Stansbury, Version 6.11?, January 6, 1983
                        0000     .2 ;               Change the IvSSRq error return code to BadParam. This is
                        0000     .3 ;               to make it consistent with the Design Guide.
                        0000     .4 ;
                        0000     .5 ;       03-01   Jack Stansbury, Version 6.9, July 25, 1982
                        0000     .6 ;               Changed the real SYSFAO module to work in the Diagnostic
```

```
0000     .7 ;                  Supervisor environment. This includes changing the PSECT names,
0000     .8 ;                  changing a return error code, and changing the Handler routine
0000     .9 ;                  code.
0000    .10 ;
0000     46 ;          V03-005 MSH0001         Maryann S. Hinden       20-NOV-1981
0000     47 ;                  Use longword displacement to reference EXE$SIGTORET.
0000     48 ;
0000     49 ;          V03-004 DWTC001         David W. Thiel          06-Nov-1981
0000     50 ;                  Fixed condition handler.  Check argument to $ASCTIM to
0000     51 ;                  prevent exception in $ASCTIM.
0000     52 ;
0000     53 ;          V03-003 PCA0001         Paul C. Anagnostopoulos         22-Jul-1981
0000     54 ;                  Fixed a bug wherein !AF did not replace unprintable
0000     55 ;                  characters if it encountered result string overflow.
0000     56 ;                  Now it replaces those characters that it does copy.
0000     57 ;
0000     58 ;          V03-002 TCM0001         Trudy C. Matthews       10-Mar-1981
0000     59 ;                  Change CALLS with word displacement to CALLS with longword
0000     60 ;                  displacement.
0000     61 ;
0000     62 ;          V03-001 TMH0001         Tim Halvorsen   24-Feb-1981
0000     63 ;                  Add condition handler to catch access violations
0000     64 ;                  and the like, so that services like $PUTMSG do
0000     65 ;                  not cause an access violation in programs like DCL
0000     66 ;                  simply because not enough arguments were supplied.
0000     67 ;--
```

```
                          0000     .2         .SBTTL  Libraries, Equated Symbols
-2                        0000     70
                          0000     71 ;
                          0000     72 ; MACROS:
                          0000     73 ;
                          0000     74
                          0000     75         $SSDEF                          ; define system status codes
                          0000     .1 ;       $CHFDEF                         ; Condition handling facility          [01]
                          0000     .2 ;       $SFDEF                          ; Call frame definitions               [01]
-2                        0000     78
                          0000     79 ;
                          0000     80 ; EQUATED SYMBOLS:
                          0000     81 ;
                          0000     82
                00000000  0000     83         ARGCOUNT = 0                    ; offset to argument count
                00000004  0000     84         INDSC    = 4                    ; offset to input string descriptor
                00000008  0000     85         OUTLEN   = 8                    ; offset to output length
                0000000C  0000     86         OUTDSC   = 12                   ; offset to output buffer descriptor
                00000010  0000     87         FIRSTARG = 16                   ; offset to first conversion param
                          0000     88
                FFFFFFF0  0000     89         INLEN    = -16                  ; local offset to input length remaining
                FFFFFFF4  0000     90         INPTR    = -12                  ; local offset to input string pointer
                FFFFFFF8  0000     91         LASTVAL  = -8                   ; local offset to last value converted
                FFFFFFFC  0000     92         FIELDEND = -4                   ; local offset to end of defined field
                          0000     93
                0000000D  0000     94         CR       = 13                   ; carriage return
                0000000A  0000     95         LF       = 10                   ; line feed
                00000021  0000     96         EXCL     = 33                   ; exclamation ('!')
                00000009  0000     97         TAB      = 9                    ; horizontal tab
                0000000C  0000     98         FF       = 12                   ; form feed
```

```
                                0000    99
                                0000   100 ;
                                0000   101 ; OWN STORAGE:
                                0000   102 ;
                                0000   103
                            00000000     .1         .PSect  Data    NoExe, Shr, NoWrt, Long ;                    [01]
-1                              0000   105
                                0000   106 ASC_NAMES:
                  35 34 33 32 31 30  0000   107         .ASCII  /0123456789ABCDEF/      ; ASCII digits
                  42 41 39 38 37 36  0006
                        46 45 44 43  000C
                                0010   108
                                0010   109 ;
                                0010   110 ;       The following table contains the first character for all
                                0010   111 ;       FAO conversion directives.  The first part of the table
                                0010   112 ;       contains the first character for two-character directives,
                                0010   113 ;       while the second half of the table contains the one-character
                                0010   114 ;       directives.
                                0010   115 ;
                                0010   116 ;       NOTE -- The ordering of this table must be preserved.  The index
                                0010   117 ;               of the directives found in this table is used to dispatch
                                0010   118 ;               via a CASE statement in the main program (FAO).
                                0010   119 ;               Routine CVTNUM also uses the index to dispatch and to
                                0010   120 ;               compute the proper radix for the conversion.
                                0010   121 ;
                                0010   122
                                0010   123 CNTRL_TABLE:
                                0010   124 TWO_CHAR_CNTRLS:
                             4F 0010   125         .ASCII  /O/                     ; octal conversions
                             58 0011   126         .ASCII  /X/                     ; hex conversions
                             55 0012   127         .ASCII  /U/                     ; unsigned decimal
                             53 0013   128         .ASCII  /S/                     ; signed decimal
                             5A 0014   129         .ASCII  /Z/                     ; unsigned decimal zero filled
                             41 0015   130         .ASCII  /A/                     ; ascii insertion directives
                             25 0016   131         .ASCII  /%/                     ; time conversion, or plural indication
                             2A 0017   132         .ASCII  /*/                     ; character repeater
                                0018     .1
                                0018   133 ONE_CHAR_CNTRLS:
                             2B 0018   134         .ASCII  /+/                     ; skip argument
                             2D 0019   135         .ASCII  /-/                     ; backup argument
                             3C 001A   136         .ASCII  /</                     ; begin field definition
                             3E 001B   137         .ASCII  />/                     ; end of field definition
                                001C     .1
                                001C   138 REPLACE_CHRS:                           ; these are one or two char replacements
                             2F 001C   139         .ASCII  ./.                     ; newline
                             5F 001D   140         .ASCII  / /                     ; tab
                             5E 001E   141         .ASCII  /^/                     ; form feed
                             21 001F   142         .ASCII  /!/                     ; insert exclamation
                                0020     .1
                       00000010 0020   143 CNTRL_LENGTH = .-CNTRL_TABLE            ; length of table
                                0020   144
                       00000008 0020   145 ONECHAR_INDEX = CNTRL_LENGTH - <ONE_CHAR_CNTRLS - CNTRL_TABLE>
                                0020   146
                       0000000C 0020   147 REPL_OFFSET = REPLACE_CHRS - CNTRL_TABLE ; offset of replacement chars
                                0020   148
                                0020   149 STRING_TYPES:
                  46 44 53 43   0020   150         .ASCII  /CSDF/                  ; ascii string types
```

```
                         0024   151 DATA_TYPES:
                   4C 57 42  0024   152         .ASCII  /BWL/                   ; byte, word , or long
                         0027   153 PERCENT_STR:
                   54 44 53  0027   154         .ASCII  /SDT/                   ; subtypes for % directive
                         002A   155 FIELDS:
                   20 10 08  002A   156         .BYTE   8,16,32                 ; field size for B,W,and L
                         002D   157 REPLACEMENT:
                21 0C 09 0A  002D   158         .BYTE   LF,TAB,FF,EXCL          ; simple replacement table
                         0031   159
                         0031   160 ;
                         0031   161 ; The following array contains the number of Octal and Hex digits in
                         0031   162 ; byte , word, and longword fields.  The byte digits are first, the
                         0031   163 ; hex digits starting at the 4'th entry so that the array may be
                         0031   164 ; context indexed.
                         C031   165 ;
                         0031   166
                         0031   167 OCT_HEX_DIGITS:
                00 08 06 03  0031   168         .BYTE   3,6,11,0
                   08 04 02  0035   169         .BYTE   2,4,8
                         0038   170
                         0038   171 RADIX:
             0A 0A 0A 10 08  0038   172         .BYTE   8,16,10,10,10           ; radix for numeric conversisons
```

```
                     003D    .2          .SBTTL  FAO - Main program
                00000000    .3          .PSect  Code    Exe, Shr, NoWrt, Long   ;                                   [01]
-4                   0000  177 ;++
                     0000  178 ; FUNCTIONAL DESCRIPTION:
                     0000  179 ;
                     0000  180 ;       This routine is the entry point for the FAO and FAOL system
                     0000  181 ;       services.   The caller's control string is scanned for control
                     0000  182 ;       characters ('!').  All other information is simply passed to
                     0000  183 ;       the output buffer.  If a control directive is found, it is parsed
                     0000  184 ;       and an action routine is dispatched.
                     0000  185 ;
                     0000  186 ; CALLING SEQUENCE:
                     0000  187 ;
                     0000  188 ;       CALLS or CALLG  to SYS$FAO or SYS$FAOL
                     0000  189 ;
                     0000  190 ; INPUT PARAMETERS:
                     0000  191 ;
                     0000  192 ;       INDSC   - The address of a string descriptor for the input
                     0000  193 ;                 control string.
                     0000  194 ;       OUTLEN  - The address of a word to receive the length of
                     0000  195 ;                 the output string
                     0000  196 ;       OUTDSC  - The address of a string descriptor for the output
                     0000  197 ;                 buffer.
                     0000  198 ;       FIRSTARG - For FAOL , this is the address of a list of longword
                     0000  199 ;                  parameters.  For FAO , this is the first of a
                     0000  200 ;                  variable number of parameters which
                     0000  201 ;                  may have been passed on the call argument list.
                     0000  202 ;
                     0000  203 ; IMPLICIT INPUTS:
                     0000  204 ;
                     0000  205 ;       none
                     0000  206 ;
                     0000  207 ; OUTPUT PARAMETERS:
                     0000  208 ;
                     0000  209 ;       OUTLEN  - Word pointed to will receive length of output buffer.
                     0000  210 ;
                     0000  211 ; IMPLICIT OUTPUTS:
                     0000  212 ;
                     0000  213 ;       none
                     0000  214 ;
                     0000  215 ; COMPLETION CODES:
                     0000  216 ;
                     0000  217 ;       SS$_NORMAL      - success code, normal return
                     0000   .1 ;       SS$_BUFFEROVF   - output buffer overflow, attempt to write past end
                     0000   .2 ;                         of output buffer
                     0000   .3 ;       SS$_BadParam    - Invalid directive specified or unable to read     [02]
                     0000   .4 ;                         argument list or address arguments                  [02]
-3                   0000  221 ;
                     0000  222 ; SIDE EFFECTS:
                     0000  223 ;
                     0000  224 ;       none
                     0000  225 ;
                     0000  226 ;--
```

```
                                0000     .2
                                0000     .3 ;+
-2                              0000    229 ; Global register usage:
                                0000    230 ;
                                0000    231 ;       R7,R8 - scratch registers
                                0000    232 ;       R9  - number of characters remaining in output buffer
                                0000    233 ;       R10 - current position in output buffer
                                0000    234 ;       R11 - pointer to next conversion parameter
                                0000    235 ;
                                0000    236 ; Locals
                                0000    237 ;
                                0000    238 ;       INLEN(FP) - (word) length of input control string
                                0000    239 ;       INPTR(FP) - address of position in input control string
                                0000     .1 ;-
-1                              C000    241
                                0000    242 ;
                                0000    243 ; Entry point for call with multiple arguments on stack
                                0000    244 ;
                                0000    245
                                0000    246 EXE$FAO::
                                0000    247
                          0FFC  0000    248         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>     ; save all registers
                                0002     .1 ;       MOVAB   W^HANDLER,(FP)          ; Establish condition handler           [01]
-1                  5B  10 AC   DE  0002    250         MOVAL   FIRSTARG(AP),R11        ; get address of first argument
                          06    11  0006    251         BRB     FAO                     ; go to main routine
                                0008    252
                                0008    253 ;
                                0008    254 ; Entry point for FAOL call.
                                0008    255 ;
                                0008    256
                                0008    257 EXE$FAOL::
                                0008    258
                          0FFC  0008    259         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                000A     .1 ;       MOVAB   W^HANDLER,(FP)          ; Establish condition handler           [01]
                    5B  10 AC   D0  000A     .2         MOVL    FIRSTARG(AP),R11        ; address of first argument
```

```
                                000E     .4
-2                              000E   262 FAO:
                     7E   7C    000E   263         CLRQ    -(SP)                           ; save space for LASTVAL and FIELDEND
                  7E   04 BC    7D 0010   264         MOVQ    @INDSC(AP),-(SP)            ; save locals on stack
                  59   0C BC    7D 0014   265         MOVQ    @OUTDSC(AP),R9              ; load output descriptor into R9,R10
                     59   59    3C 0018   266         MOVZWL  R9,R9                       ; ensure word length
                                001B   267
                                001B   268 ;
                                001B   269 ; Look for a control character in the input string.  Copy text
                                001B   270 ; up to the control, if any , to the output buffer.
                                001B   271 ;
                                001B   272
                                001B   273 MAIN_SCAN:
                     7E   D4    001B   274         CLRL    -(SP)                           ; indicate control not found
                  F0 AD   21    3A 001D   275         LOCC    #EXCL,INLEN(FP),@INPTR(FP) ; search for control char
                        F4 BD           0021
                     02   13    0023   276         BEQL    10$                             ; branch if not found
                     6E   D6    0025   277         INCL    (SP)                            ; set indicator to show char. found
                                0027   278 10$:
                  F0 AD   50    A3 0027   279         SUBW3   R0,INLEN(FP),R6             ; calculate bytes to move
                           56           002B
                  F0 AD   50    D0 002C   280         MOVL    R0,INLEN(FP)                ; update input length remaining
                     59   56    A2 0030   281         SUBW    R6,R9                       ; update and test output length
                     75   19    0033   282         BLSS    OVERFLOW                    ; not enough room, error exit
                  F4 BD   56    28 0035   283         MOVC3   R6,@INPTR(FP),(R10)         ; move text part of input string
                           6A           0039
                        76 8E    E9 003A   284         BLBC    (SP)+,DONE                  ; leave if no controls left
                  F4 AD   51    D0 003D   285         MOVL    R1,INPTR(FP)                ; update input address pointer
                     5A   53    D0 0041   286         MOVL    R3,R10                      ; update output address pointer
                     7C   10    0044   287         BSBB    GETCHAR                     ; skip control char
```

```
-1                                      0046   289 ;
                                        0046   290 ; Parse the directive which has been found in the input string.  Set
                                        0046   291 ; up: R0 = remaining count in CNTRL_TABLE
                                        0046   292 ;     R4 = second char if two-char directive
                                        0046   293 ;     R5 = repeat count
                                        0046   294 ;     R6 = field width
                                        0046   295 ;
                                        0046   296
                                        0046   297 PARSE_DIRECTIVE:
                                        0046   298
                    55   01   D0       0046   299          MOVL    #1,R5                       ; default repeat count is 1
                         52   D4       0049   300          CLRL    R2                          ; paren indicator ( not found yet )
                       0081   30       004B   301          BSBW    GETCOUNT                    ; pull off count, if any
                         72   10       004E   302          BSBB    GETCHAR                     ; get next char from input string
                    53   28   91       0050   303          CMPB    #^A/(/,R3                   ; was next char a paren?
                         0D   12       0053   304          BNEQ    20$                         ; branch if not
                         52   D6       0055   305          INCL    R2                          ; set paren found indicator
                         56   D5       0057   306          TSTL    R6                          ; was there a repeat count?
                         03   19       0059   307          BLSS    10$                         ; no..use default
                    55   56   D0       005B   308          MOVL    R6,R5                       ; else get repeat count
                                        005E   309 10$:
                         6F   10       005E   310          BSBB    GETCOUNT                    ; look for field width
                         60   10       0060   311          BSBB    GETCHAR                     ; get next char
                                        0062   312 20$:
                    10   53   3A       0062   313          LOCC    R3,#CNTRL_LENGTH,CNTRL_TABLE ; check character in table
                00000010'EF             0065
                         39   13       006A   314          BEQL    ILLEGAL                     ; illegal directive exit
                    08   50   D1       006C   315          CMPL    R0,#ONECHAR_INDEX           ; is this a one char directive?
                         05   15       006F   316          BLEQ    30$                         ; yes, don't need any more
                         4F   10       0071   317          BSBB    GETCHAR                     ; get second control char
                    54   53   D0       0073   318          MOVL    R3,R4                       ; move to R4 for return
                                        0076   319 30$:
                      02 52   E9       0076   320          BLBC    R2,40$                      ; skip if no paren found
                         47   10       0079   321          BSBB    GETCHAR                     ; else skip paren char
                                        007B   322 40$:
                                        007B   323
                53   10   50   C3       007B   324          SUBL3   R0,#CNTRL_LENGTH,R3         ; compute offset for case table
                                        007F   325
                                        007F   326 ;
                                        007F   327 ; The following  does a BSBB to the case dispatch
                                        007F   328 ; table.  The service routines do an RSB and return into CASE_LOOP.
                                        007F   329 ;
                                        007F   330
                         02   11       007F   331          BRB     CASE_LOOP                   ; start processing loop
                                        0081   332 CASE_BSB:
                         05   10       0081   333          BSBB    FAO_CASE                    ; dispatch next directive
                                        0083   334 CASE_LOOP:
                      FB 55   F4       0083   335          SOBGEQ  R5,CASE_BSB                 ; repeat as specified
                         93   11       0086   336          BRB     MAIN_SCAN                   ; else continue string processing
```

```
-1                                      0088   338 ;
                                        0088   339 ; Here is the main dispatch table for dispatching FAO service
                                        0088   340 ; routines.  The case is entered via BSBB from CASE_BSB. The routines
                                        0088   341 ; RSB to CASE_LOOP.  Since the 5 numeric conversion directives all
                                        0088   342 ; dispatch to the same routine, the case has a base of 5 and the
                                        0088   343 ; numeric directives fall through to the statement following the CASE.
                                        0088   344 ;
                                        0088   345 ; Registers R0, R1, and R2 may be scratched by service routines.
                                        0088   346 ;
                                        0088   347
                                        0088   348 FAO_CASE:
                                        0088   349         CASE    R3,<-                   ; dispatch to service routine
                                        0088   350                 CVTASC,-                ; ascii string insertion
                                        0088   351                 PERCENT,-               ; insert ascii time or plural 'S'
                                        0088   352                 REPEATIT,-              ; repeat character 'n' times
                                        0088   353                 INCR_ARGPTR,-           ; skip next parameter
                                        0088   354                 DECR_ARGPTR,-           ; backup to previous parameter
                                        0088   355                 STARTFIELD,-            ; define fixed length field
                                        0088   356                 ENDFIELD,-              ; terminate fixed length field
                                        0088   357                 NEWLINE,-               ; insert CR/LF
                                        0088   358                 INSERT_CHAR,-           ; insert TAB
                                        0088   359                 INSERT_CHAR,-           ; insert form feed
                                        0088   360                 INSERT_CHAR,-           ; insert '!'
                                        0088   361         >,B,#5                          ; offset start by 5
                                        00A2   362
                        00D3    31      00A2   363         BRW     CVTNUM                  ; dispatch to numeric conversion
                                        00A5   364
                                        00A5   365
                                        00A5   366 ILLEGAL:
                    50  14    3C        00A5    .1         MOVZWL  #SS$_BadParam,R0        ; Error return code                    [02]
                        0C    11        00A8    .2         BRB     FAO_EXIT
                                        00AA    .3
-2                                      00AA   369 OVERFLOW:
                50  0601  8F    3C      00AA   370         MOVZWL  #SS$_BUFFEROVF,R0       ; error return code
                        59    D4        00AF   371         CLRL    R9                      ; ensure correct return length
                        03    11        00B1   372         BRB     FAO_EXIT
                                        00B3    .1
                                        00B3    .2 DONE:
                    50  01    3C        00B3    .3         MOVZWL  #SS$_NORMAL,R0          ; no errors
                                        00B6    .4
-2                                      00B6   375 FAO_EXIT:
                      08 AC   D5        00B6   376         TSTL    OUTLEN(AP)              ; was a return length required?
                        06    13        00B9   377         BEQL    10$                     ; branch if not
                  0C BC   59    A3      00BB   378         SUBW3   R9,@OUTDSC(AP),@OUTLEN(AP) ; compute and return output buffer length
                      08 BC             00BF
                                04      00C1   379 10$:     RET
```

```
-1                              00C2   381          .SBTTL  GETCHAR - Routine to get next char from input string
                                00C2   382
                                00C2   383 ;++
                                00C2   384 ;
                                00C2   385 ; FUNCTIONAL DESCRIPTION:
                                00C2   386 ;
                                00C2   387 ;       This routine gets the next character from the input control
                                00C2   388 ;       string, updating the length and address pointers.  If the length
                                00C2   389 ;       goes negative, an error exit is called.
                                00C2   390 ;
                                00C2   391 ; CALLING SEQUENCE:
                                00C2   392 ;
                                00C2   393 ;       JSB (R8)
                                00C2   394 ;
                                00C2   395 ; INPUT PARAMETERS:
                                00C2   396 ;
                                00C2   397 ;       none
                                00C2   398 ;
                                00C2   399 ; IMPLICIT INPUTS:
                                00C2   400 ;
                                00C2   401 ;       INLEN(FP) -  lower word has remaining length of input string
                                00C2   402 ;       INPTR(FP) -  is pointer to current string position
                                00C2   403 ;
                                00C2   404 ; OUTPUTS:
                                00C2   405 ;
                                00C2   406 ;       R3 - next character in input string
                                00C2   407 ;
                                00C2   408 ; IMPLICIT OUTPUTS:
                                00C2   409 ;
                                00C2   410 ;       none
                                00C2   411 ;
                                00C2   412 ; COMPLETION CODES:
                                00C2   413 ;
                                00C2   414 ;       none
                                00C2   415 ;
                                00C2   416 ; SIDE EFFECTS:
                                00C2   417 ;
                                00C2   418 ;       input pointers on stack are updated
                                00C2   419 ;       error may cause jump to ILLEGAL
                                00C2   420 ;--
```

```
-1                                    00C2   422 GETCHAR:
                        F0 AD    B7   00C2   423          DECW    INLEN(FP)                ; decr input length remaining
                           DE    19   00C5   424          BLSS    ILLEGAL                  ; error if no more left
                     53    F4 BD    9A   00C7   425          MOVZBL  @INPTR(FP),R3            ; get next character
                        F4 AD    D6   00CB   426          INCL    INPTR(FP)                ; update pointer
                                  05   00CE   427          RSB                              ; return
```

```
-1                              00CF    429          .SBTTL  GETCOUNT - Routine to get repeat-count or field-width
                                00CF    430
                                00CF    431 ;++
                                00CF    432 ;
                                00CF    433 ; FUNCTIONAL DESCRIPTION:
                                00CF    434 ;
                                00CF    435 ;       This subroutine to PARSE_DIRECTIVE scans for a repeat-count or
                                00CF    436 ;       field-width in the directive in the input stream.  If a numeric
                                00CF    437 ;       count is found, it is converted to binary.  If a '#' character
                                00CF    438 ;       is found, the count is taken from the next parameter
                                00CF    439 ;       in the parameter list.
                                00CF    440 ;
                                00CF    441 ; CALLING SEQUENCE:
                                00CF    442 ;
                                00CF    443 ;       JSB or BSB
                                00CF    444 ;
                                00CF    445 ; INPUTS:
                                00CF    446 ;
                                00CF    447 ;       R11     - parameter pointer
                                00CF    448 ;
                                00CF    449 ; IMPLICIT INPUTS:
                                00CF    450 ;
                                00CF    451 ;       none
                                00CF    452 ;
                                00CF    453 ; OUTPUTS:
                                00CF    454 ;
                                00CF    455 ;       R6      - value of count, if # or number found, else -1
                                00CF    456 ;
                                00CF    457 ; IMPLICIT OUTPUTS:
                                00CF    458 ;
                                00CF    459 ;       R11 may be modified if a parameter is taken from the stack
                                00CF    460 ;
                                00CF    461 ; COMPLETION CODES:
                                00CF    462 ;
                                00CF    463 ;       none
                                00CF    464 ;
                                00CF    465 ; SIDE EFFECTS:
                                00CF    466 ;
                                00CF    467 ;       R1, R3, and R4 are destroyed
                                00CF    468 ;--
```

```
-2
                                00CF   471
                                00CF   472 GETCOUNT:
                    56   01   CE 00CF   473          MNEGL   #1,R6                   ; not found indicator
                F4  BD   23   91 00D2   474          CMPB    #^A/#/,@INPTR(FP)       ; is this a param. count?
                         26   13 00D6   475          BEQL    40$                     ; yes .. pull next param
                         53   7C 00D8   476          CLRQ    R3                      ; zero buffer for digit (R3)
                                00DA   477                                          ; ... and accumulator for sum (R4)
                51   F4  AD   D0 00DA   478          MOVL    INPTR(FP),R1            ; remember where we were
                                00DE   479 10$:
                F4  BD   30   83 00DE   480          SUBB3   #^A/0/,@INPTR(FP),R3    ; subtract ascii 0 from char
                         53        00E2
                         0F   19 00E3   481          BLSS    20$                     ; branch if not numeric
                    53   09   91 00E5   482          CMPB    #^A/9/-^A/0/,R3         ; still numeric?
                         0A   19 00E8   483          BLSS    20$                     ; no, branch
                    54   0A   C4 00EA   484          MULL2   #10,R4                  ; shift for next digit
                    54   53   C0 00ED   485          ADDL    R3,R4                   ; add in next digit
                         D0   10 00F0   486          BSBB    GETCHAR                 ; skip digit we took
                         EA   11 00F2   487          BRB     10$                     ; continue while numeric
                                00F4   488 20$:
                F4  AD   51   D1 00F4   489          CMPL    R1,INPTR(FP)            ; did we get any chars?
                         03   13 00F8   490          BEQL    30$                     ; no, leave
                    56   54   D0 00FA   491          MOVL    R4,R6                   ; yes, return value
                                00FD   492 30$:
                              05 00FD   493          RSB                             ; return
                                00FE   494
                                00FE   495 40$:
                    56   8B   D0 00FE   496          MOVL    (R11)+,R6               ; get value from next parameter
                         BF   10 0101   497          BSBB    GETCHAR                 ; skip '#'
                              05 0103   498          RSB                             ; return
```

```
-1                              0104    500          .SBTTL  CVTASC  - Insert ASCII string
                                0104    501          .LIST MEB
                                0104    502
                                0104    503 ;++
                                0104    504 ;
                                0104    505 ; FUNCTIONAL DESCRIPTION:
                                0104    506 ;
                                0104    507 ;       Service routine to handle ASCII string insertions.
                                0104    508 ;       Strings are specified by several different methods.  For
                                0104    509 ;       filled strings (AF) , non-printing characters are output
                                0104    510 ;       as dots ('.').
                                0104    511 ;
                                0104    512 ; CALLING SEQUENCE:
                                0104    513 ;
                                C104    514 ;       JSB or BSB
                                0104    515 ;
                                0104    516 ; INPUTS:
                                0104    517 ;
                                0104    518 ;       R3      - index of first control char in CNTRL_TABLE
                                0104    519 ;       R4      - second control character
                                0104    520 ;       R6      - output field width
                                0104    521 ;       R9      - output buffer length remaining
                                0104    522 ;       R10     - output buffer pointer
                                0104    523 ;       R11     - parameter pointer
                                0104    524 ;
                                0104    525 ; IMPLICIT INPUTS:
                                0104    526 ;
                                0104    527 ;       none
                                0104    528 ;
                                0104    529 ; OUTPUTS:
                                0104    530 ;
                                0104    531 ;       none
                                0104    532 ;
                                0104    533 ; IMPLICIT OUTPUTS:
                                0104    534 ;
                                0104    535 ;       R9 and R10 are update to point to current position in output buffer
                                0104    536 ;       R11 is updated as parameters are taken from the stack
                                0104    537 ;
                                0104    538 ; ROUTINE VALUE:
                                0104    539 ;
                                0104    540 ;       none
                                0104    541 ;
                                0104    542 ; SIDE EFFECTS:
                                0104    543 ;
                                0104    544 ;       R7 and R8 are destroyed
                                0104    545 ;--
```

```
                                0104     .2
                                0104     .3 CVTASC:
-4                      0078 8F   BB    0104    550          PUSHR   #^M<R3,R4,R5,R6>        ; save registers
                             57   D4    0108    551          CLRL    R7                      ; set filled indicator to not filled
                          04 54   3A    010A    552          LOCC    R4,#4,STRING_TYPES      ; search for string subtype
                00000020'EF             010D
                             61   13    0112    553          BEQL    110$                    ; error if not found
                                0114    554
                                0114    555 ;
                                0114    556 ;       R0 = 1 - filled , 2 - 2 arg desc. , 3 - str. desc. , 4 - cstring
                                0114    557 ;
                                0114    558          CASE    R0,<10$,20$,30$>,B,#2   ; case on descriptor type, base = 2
                      02'  02   50   8F  0114                   CASEB   R0,#2,S^#<<30003$-30002$>/2>-1
                                0118         30002$:
                           0008' C118                .SIGNED_WORD    10$-30002$
                           000D' 011A                .SIGNED_WORD    20$-30002$
                           0015' 011C                .SIGNED_WORD    30$-30002$
                                011E         30003$:
                                011E    559
                                011E    560 ;
                                011E    561 ; Case falls through here for filled ascii strings. Two argument
                                011E    562 ; descriptor is used.
                                011E    563 ;
                                011E    564
                             57   D6    011E    565          INCL    R7                      ; set filled indicator for filled ascii
                                0120    566 10$:
                          51 8B   7D    0120    567          MOVQ    (R11)+,R1               ; get length and address
                             0E   11    0123    568          BRB     40$                     ; continue
                                0125    569
                                0125    570 ;
                                0125    571 ; Standard system string descriptor
                                0125    572 ;
                                0125    573
                                0125    574 20$:
                       51   9B   7D    0125    575          MOVQ    @(R11)+,R1              ; move descriptor to R1,R2
                       51   51   3C    0128    576          MOVZWL  R1,R1                   ; make sure length is word
                            06   11    012B    577          BRB     40$                     ; continue
                                012D    578
                                012D    579 ;
                                012D    580 ; Ascii counted string, first byte contains length
                                012D    581 ;
                                012D    582
                                012D    583 30$:
                       52   8B   D0    012D    584          MOVL    (R11)+,R2               ; address of counted string
                       51   82   9A    0130    585          MOVZBL  (R2)+,R1                ; get length and skip byte count
                                0133    586
                                0133    587 40$:
                                0133    588
                                0133    589 ;
                                0133    590 ; Here, R1 has string length, R2 has string address. Check length against
                                0133    591 ; specified field width to decide how much string to move.
                                0133    592 ;
                                0133    593
                       58   56   D0    0133    594          MOVL    R6,R8                   ; was a width specified?
                            03   18    0136    595          BGEQ    50$                     ; branch if so
                       5.   51   D0    0138    596          MOVL    R1,R8                   ; if not, use string length instead
                                013B    597 50$:
```

```
                                013B   598
                                013B   599 ;
                                013B   600 ; The string is moved to the output buffer with blank fill at the
                                013B   601 ; end.  The output pointers are then updated by the field width, so
                                013B   602 ; that the string will be truncated if it was longer than the field
                                013B   603 ; width.  If the string is filled, a second pass is made to change
                                013B   604 ; non-printing characters to dots.
                                013B   605 ;
                                013B   606
          56    59    D0      013B   607          MOVL    R9,R6                       ; copy remaining char count
                                013E   608                                          ; NOTE we have to use R6 here.
          59    58    C2      013E   609          SUBL    R8,R9                       ; update length remaining
                      03  19    0141   610          BLSS    55$                         ; Overflow, use remaining length.
          56    58    D0      0143   611          MOVL    R8,R6                       ; else move only required length
    20    62    51    2C      0146   612 55$:     MOVC5   R1,(R2),#^A/ /,R6,(R10) ; move string, fill at end
          6A    56            014A
                                014C   613
          52    5A    D0      014C   614          MOVL    R10,R2                      ; save output address
          5A    56    C0      014F   615          ADDL    R6,R10                      ; update output pointer
             14 57    E9      0152   616          BLBC    R7,90$                      ; all done if not filled ASCII
                                0155   617 60$:                                     ; R7 will now become loop counter.
          20    62    91      0155   618          CMPB    (R2),#^040                  ; printing character?
                      06  19    0158   619          BLSS    70$                         ; no, fill with dot
       7E 8F    62    91      015A   620          CMPB    (R2),#^0176                 ; still printing?
                      03  15    015E   621          BLEQ    80$                         ; yes, skip this one
                                0160   622 70$:
          62    2E    90      0160   623          MOVB    #^A/./,(R2)                 ; insert dot in place of char
                                0163   624 80$:
          52    D6            0163   625          INCL    R2                          ; point to next character
    EC 57    56    F3      0165   626          AOBLEQ  R6,R7,60$                   ; continue until done
                                0169   627
                                0169   628 90$:
          59    D5            0169   629          TSTL    R9                          ; Did we get result overflow above?
                      05  19    016B   630          BLSS    100$                        ; Yes, branch to tell user.
       0078 8F    BA      016D   631          POPR    #^M<R3,R4,R5,R6>
                      05    0171   632          RSB                                 ; return
                                0172   633
                                0172   634 100$:
             FF35   31      0172   635          BRW     OVERFLOW
                                0175   636 110$:
             FF2D   31      0175   637          BRW     ILLEGAL
```

```
-1                              0178   639          .SBTTL  CVTNUM - Convert numeric parameter to ASCII
                                0178   640
                                0178   641 ;++
                                0178   642 ;
                                0178   643 ; FUNCTIONAL DESCRIPTION:
                                0178   644 ;
                                0178   645 ;       This routine handles the various HEX, OCTAL, and DECIMAL
                                0178   646 ;       conversions. The proper field is extracted from the
                                0178   647 ;       parameter (byte, word , or long) and the needed output
                                0178   648 ;       width is determined.  This is compared with the user
                                0178   649 ;       specified field width to determine if padding of filling
                                0178   650 ;       is needed.  The entire field with fill is built on the
                                0178   651 ;       stack and then moved so that the result will be correct
                                0178   652 ;       on buffer overflow.
                                0178   653 ;
                                0178   654 ; CALLING SEQUENCE:
                                0178   655 ;
                                0178   656 ;       JSB or BSB
                                0178   657 ;
                                0178   658 ; INPUTS:
                                0178   659 ;
                                0178   660 ;       R3      - index of directive in CNTRL_TABLE.
                                0178   661 ;                       0 = Octal
                                0178   662 ;                       1 = heX
                                0178   663 ;                       2 = Unsigned decimal
                                0178   664 ;                       3 = Signed decimal
                                0178   665 ;                       4 = Zero filled unsigned decimal
                                0178   666 ;       R4      - second char of directive (B,W, or L)
                                0178   667 ;       R6      - field width, or -1 if none
                                0178   668 ;       R9      - output length remaining
                                0178   669 ;       R10     - output position pointer
                                0178   670 ;       R11     - next parameter pointer
                                0178   671 ;
                                0178   672 ; IMPLICIT INPUTS:
                                0178   673 ;
                                0178   674 ;       none
                                0178   675 ;
                                0178   676 ; OUTPUTS:
                                0178   677 ;
                                0178   678 ;       none
                                0178   679 ;
                                0178   680 ; IMPLICIT OUTPUTS:
                                0178   681 ;
                                0178   682 ;       none
                                0178   683 ;
                                0178   684 ; ROUTINE VALUE:
                                0178   685 ;
                                0178   686 ;       none
                                0178   687 ;
                                0178   688 ; SIDE EFFECTS:
                                0178   689 ;
                                0178   690 ;       none
                                0178   691 ;--
```

```
-1                                      0178   693 ;
                                        0178   694 ; The registers will be set up as follows
                                        0178   695 ;
                                        0178   696 ;       R0  - max digits to be output
                                        0178   697 ;       R1  - 0 -> byte, 1 -> word , 2 -> long
                                        0178   698 ;       R2  - value to be converted
                                        0178   699 ;       R4  - conversion radix
                                        0178   700 ;       R5  - sign indicator, 1 -> sign to be output, 0 otherwise
                                        0178   701 ;       R7  - fill character, (blank, zero for !Z, or * on width too small
                                        0178   702 ;             for decimal conversions)
                                        0178   703 ;       R8  - total width of field to be output
                                        0178   704 ;
                                        0178   705
                                        0178   706 CVTNUM:
-1                          38   BB     C178   708             PUSHR   #^M<R3,R4,R5>
                                        017A   709
                      03    54   3A     017A   710             LOCC    R4,#3,DATA_TYPES        ; determine data type
                00000024'EF             017D
                      03    12   0182   711             BNEQ    10$                     ; continue if legal directive
                    FF1E    31   0184   712             BRW     ILLEGAL                 ; else take error condition
                                        0187   713 10$:
              51    03    50   C3     0187   714             SUBL3   R0,#3,R1                ; convert to index
                    52    8B   D0     018B   715             MOVL    (R11)+,R2               ; get next longword parameter
                          00   EF     018E   716             EXTZV   #0,FIELDS[R1],R2,R2     ; select proper field
                00000028'EF41           0190
                    52    52          0196
                          55   D4     0198   717             CLRL    R5                      ; note unsigned
                    57    20   90     019A   718             MOVB    #^A/ /,R7               ; default fill char is blank
                00000038'EF43   9A     019D   719             MOVZBL  RADIX[R3],R4            ; get conversion radix
                          54           01A4
                                       01A5   720
                                       01A5   721 ;
                                       01A5   722 ; Case on the type of conversion.  Note that base is set
                                       01A5   723 ; so that octal and hex conversions fall through case table.
                                       01A5   724 ;
                                       01A5   725
                                       01A5   726             CASE    R3,<40$,30$,20$>,,#2    ; base index of 2
        02'   02    53   AF     01A5                   CASEW   R3,#2,S^#<<30005$-30004$>/2>-1
                                       01A9           30004$:
                      003C'    01A9                   .SIGNED_WORD    40$-30004$
                      0029'    01AB                   .SIGNED_WORD    30$-30004$
                      0024'    01AD                   .SIGNED_WORD    20$-30004$
                                       01AF           30005$:
                                       01AF   727
                                       01AF   728 ;
                                       01AF   729 ; Octal and Hex fall through here
                                       01AF   730 ;
                                       01AF   731
           50     6143   DE     01AF   732             MOVAL   (R1)[R3],R0             ; compute index in OCT_HEX_DIGITS
        00000031'EF40   9A     01B3   733             MOVZBL  OCT_HEX_DIGITS[R0],R0   ; get number of digits to output
                    50          01BA
              58    56   D0     01BB   734             MOVL    R6,R8                   ; user specified width?
                    03   18     01BE   735             BGEQ    15$                     ; yes, use it as width
              58    50   D0     01C0   736             MOVL    R0,R8                   ; else take needed space
                                       01C3   737 15$:
              50    58   D1     01C3   738             CMPL    R8,R0                   ; width lss default digits?
                    47   18     01C6   739             BGEQ    60$                     ; no, fill to user specified width
```

```
                50   58   D0  01C8   740          MOVL    R8,R0                   ; else output only specified width
                     42   11  01CB   741          BRB     60$
                                01CD   742
                                01CD   743 ;
                                01CD   744 ; Unsigned decimal with zero fill
                                01CD   745 ;
                                01CD   746
                                01CD   747 20$:
                57   30   90  01CD   748          MOVB    #^A/0/,R7               ; insert new fill char
                     13   11  01D0   749          BRB     40$                     ; continue with normal dec. code
                                01D2   750
                                01D2   751 ;
                                01D2   752 ; Signed decimal conversion
                                01D2   753 ;
                                C1D2   754
                                01D2   755 30$:
                     00   EE  01D2   756          EXTV    #0,FIELDS[R1],R2,R2     ; sign extend the field
          0000002A'EF41         01D4
                52   52         01DA
             05 52   1F   E1  01DC   757          BBC     #31,R2,40$              ; not negative, continue
                     55   D6  01E0   758          INCL    R5                      ; else note that value negative
                52   52   CE  01E2   759          MNEGL   R2,R2                   ; and make it positive
                                01E5   760
                                01E5   761 40$:                                  ; common decimal processing
                                01E5   762
                                01E5   763 ;
                                01E5   764 ; Determine the number of digits needed to print number in ASCII
                                01E5   765 ; decimal representation.
                                01E5   766 ;
                                01E5   767
            50   01   D0  01E5   768          MOVL    #1,R0                   ; init digit counter
            53   54   D0  01E8   769          MOVL    R4,R3                   ; copy first power of 10
                                01EB   770 44$:
            53   52   D1  01EB   771          CMPL    R2,R3                   ; does it fit?
                 07   1F  01EE   772          BLSSU   48$                     ; yes, R0 has count if so
            53   54   C4  01F0   773          MULL    R4,R3                   ; else compute next power of ten
         F4 50   54   F2  01F3   774          AOBLSS  R4,R0,44$               ; continue (10 digits is largest possible)
                                01F7   775 48$:
      53    55   50   C1  01F7   776          ADDL3   R0,R5,R3                ; add in sign, if one exists
            58   56   D0  01FB   777          MOVL    R6,R8                   ; did user specify width?
                 05   18  01FE   778          BGEQ    50$                     ; yes, use it for field width
            58   53   D0  0200   779          MOVL    R3,R8                   ; else use amount needed
                 0A   11  0203   780          BRB     60$                     ; continue
                                0205   781 50$:
            58   53   D1  0205   782          CMPL    R3,R8                   ; is there space within specified width?
                 05   15  0208   783          BLEQ    60$                     ; yes, go on
            57   2A   90  020A   784          MOVB    #^A/*/,R7               ; no room, fill with stars
                 50   D4  020D   785          CLRL    R0                      ; output no digits
                                020F   786
                                020F   787 60$:
         F8 AD   52   D0  020F   788          MOVL    R2,LASTVAL(FP)          ; remember value to be converted
                                0213   789
                                0213   790 ;
                                0213   791 ; Insert the ASCII representation for the value in R2 into the
                                0213   792 ; output buffer.
                                0213   793 ;
                                0213   794
```

```
                                 0213   795 CVT_BIN_TO_ASC:
                                 0213   796
                0840 8F    BB    0213   797          PUSHR   #^M<R6,R11>              ; save work registers
                  04 A8    9F    0217   798          PUSHAB  4(R8)                   ; compute stack space needed for buffer
                6E   03    CA    021A   799          BICL    #3,(SP)                 ; round stack to longword
                5B   5E    D0    021D   800          MOVL    SP,R11                  ; save stack pointer
                5E   6B    C2    0220   801          SUBL    (R11),SP                ; leave buffer space on stack
                                 0223   802
                     53    D4    0223   803          CLRL    R3                      ; clear upper half of quad quotient
                     51    CE    0225   804          MNEGL   #1,R1                   ; init digit counter for loop
                     0D    11    0228   805          BRB     15$                     ; start loop
                                 022A   806 10$:
          52    52   54    7B    022A   807          EDIV    R4,R2,R2,R6             ; R2 <- quotient, R6 <- remainder
                     56          022E
00000000'EF       46    90    C22F   808          MOVB    ASC_NAMES[R6],-(R11)    ; output ascii digit
                     7B          0236
                                 0237   809 15$:
             EF 51   50    F2    0237   810          AOBLSS  R0,R1,10$               ; one more digit, done yet?
                  05 55    E9    023B   811          BLBC    R5,20$                  ; branch if no sign to output
                7B   2D    90    023E   812          MOVB    #^A/-/,-(R11)           ; output sign
                     51    D6    0241   813          INCL    R1                      ;
                                 0243   814 20$:
                                 0243   815
                                 0243   816 ;
                                 0243   817 ; If field (R8) is not full, then fill remainder with the fill character
                                 0243   818 ;
                                 0243   819
                     03    11    0243   820          BRB     40$                     ; start the loop
                                 0245   821 30$:
                7B   57    90    0245   822          MOVB    R7,-(R11)               ; insert fill character
                                 0248   823 40$:
          F9 51    58    F3    0248   824          AOBLEQ  R8,R1,30$               ; fill until full
                                 024C   825
                                 024C   826 ;
                                 024C   827 ; Now copy stack back to buffer, checking for overflow
                                 024C   828 ;
                                 024C   829
                     08    11    024C   830          BRB     70$                     ; start loop
                                 024E   831 50$:
                  02 59    F4    024E   832          SOBGEQ  R9,60$                  ; update length, check for overflow
                     21    11    0251   833          BRB     INSERT_OVF              ; handle overflow
                8A   8B    90    0253   834 60$:     MOVB    (R11)+,(R10)+           ; move char to output buffer
                  F5 58    F4    0256   835 70$:     SOBGEQ  R8,50$                  ; move entire string
                                 0259   836
                                 0259   837 ;
                                 0259   838 ; Now clean up mess on stack
                                 0259   839 ;
                                 0259   840
                5E   5B    D0    0259   841          MOVL    R11,SP                  ; restore stack
                0841 8F    BA    025C   842          POPR    #^M<R0,R6,R11>          ; remove top of stack and restore regs
                                 0260   843
                                 0260   844 ;
                                 0260   845 ; Restore registers and return from service routine.
                                 0260   846 ;
                                 0260   847
                     38    BA    0260   848          POPR    #^M<R3,R4,R5>
                           05    0262   849          RSB
```

```
-1                              0263    851         .SBTTL  QUICKSERVE - Small service routines
                                0263    852
                                0263    853 ;++
                                0263    854 ;
                                0263    855 ; FUNCTIONAL DESCRIPTION:
                                0263    856 ;
                                0263    857 ;       Following are a collection of short service routines for
                                0263    858 ;       FAO directives.
                                0263    859 ;
                                0263    860 ; CALLING SEQUENCE:
                                0263    861 ;
                                0263    862 ;       JSB or BSB
                                0263    863 ;
                                0263    864 ; INPUTS:
                                0263    865 ;
                                0263    866 ;       R3  - index in CNTRL_TABLE of the directive
                                0263    867 ;       R4  - second character of two-char directive, if any
                                0263    868 ;       R6  - user specified field width, if any (ignored for singal char
                                0263    869 ;               and argument directives)
                                0263    870 ;       R9  - output length remaining
                                0263    871 ;       R10 - output position pointer
                                0263    872 ;
                                0263    873 ; IMPLICIT INPUTS:
                                0263    874 ;
                                0263    875 ;       none
                                0263    876 ;
                                0263    877 ; OUTPUTS:
                                0263    878 ;
                                0263    879 ;       none
                                0263    880 ;
                                0263    881 ; IMPLICIT OUTPUTS:
                                0263    882 ;
                                0263    883 ;       R9 and R10 are modified
                                0263    884 ;
                                0263    885 ; COMPLETION CODES:
                                0263    886 ;
                                0263    887 ;       none
                                0263    888 ;
                                0263    889 ; SIDE EFFECTS:
                                0263    890 ;
                                0263    891 ;       none
                                0263    892 ;--
```

```
                                0263     .2
                                0263     .3 INCR_ARGPTR:
-4                              0263    897 ;
                                0263    898 ; Directive to skip next parameter in parameter list
                                0263    899 ;
                                0263    900
                          8B    D5  0263    901         TSTL    (R11)+                  ; skip next parameter
                              05    0265    902         RSB                             ; exit
                                0266    903
                                0266    904 DECR_ARGPTR:
-1                              0266    906 ;
                                0266    907 ; Directive to back up and reuse last parameter in parameter list
                                0266    908 ;
                                0266    909
                          7B    D5  0266    910         TSTL    -(R11)                  ; back up argument pointer
                              05    0268    911         RSB                             ; exit
                                0269    912
                                0269    913 NEWLINE:
-1                              0269    915 ;
                                0269    916 ; Insert carriage return, line feed into output buffer
                                0269    917 ;
                                0269    918
                       02 59    F4  0269    919         SOBGEQ  R9,10$                  ; room for CR?, branch if so
                          06    11  026C    920         BRB     INSERT_OVF              ; no room in output buffer
                                026E    921 10$:
                    8A    0D    90  026E    922         MOVB    #CR,(R10)+              ; insert CR in output buffer
                                0271    923                                             ; continue for LF insertion
                                0271    924
                                0271    925 INSERT_CHAR:
-1                              0271    927 ;
                                0271    928 ; Make simple one character insertion in the output buffer.
                                0271    929 ;
                                0271    930
                       03 59    F4  0271    931         SOBGEQ  R9,INSERT_IT            ; check length, branch if ok
                                0274     .1
                                0274     .2 INSERT_OVF:
                      FE33      31  0274     .3         BRW     OVERFLOW                ; error , no room in output buffer
                                0277     .4
                                0277     .5 INSERT_IT:
-4                              0277    936 ;
                                0277    937 ; Insert the character by computing the index into the replacement table
                                0277    938 ;
                                0277    939
          00000021'EF43         90  0277    940         MOVB    REPLACEMENT-REPL_OFFSET[R3],(R10)+ ; insert the char
                        8A              027E
                              05    027F    941         RSB
                                0280    942
                                0280    943 ;
                                0280    944 ; Directive to repeat a particular character 'n' times, where 'n' is
                                0280    945 ; specified by the field width in the directive.
                                0280    946 ;
                                0280    947
                                0280    948 REPEATIT:
                          38    BB  0280    949         PUSHR   #^M<R3,R4,R5>           ; save regs for MOVC5 clobber
                          56    D5  0282    950         TSTL    R6                      ; check if width was specified
                          15    19  0284    951         BLSS    ILLFIELD                ; illegal if none specified
                    59    56    C2  0286    952         SUBL    R6,R9                   ; compute remaining output length
```

```
                E9  19  0289   953          BLSS    INSERT_OVF                      ; not enough room, error
          54    6E  00  2C  028B   954          MOVC5   #0,(SP),R4,R6,(R10)     ; fill with specified character
                6A  56          028F
                5A  56  C0  0291   955          ADDL    R6,R10                  ; update output pointer
                    38  BA  0294   956          POPR    #^M<R3,R4,R5>           ; restore regs
                        05  0296   957          RSB
                            0297   958
                            0297   959 ;
                            0297   960 ; The following are the directives which define a fixed length field.
                            0297   961 ; The field width is specified with the define field directive.  At the
                            0297   962 ; end field directive, any of the field remaining is blank filled, else
                            0297   963 ; the field is truncated to the specified length.
                            0297   964 ;
                            0297   965
                            C297   966 STARTFIELD:
                    56  D5  0297   967          TSTL    R6                      ; did user specify field (must be specified)
                    03  18  0299   968          BGEQ    STARTOK                 ; yes, continue
                            029B    .1
                            029B    .2 ILLFIELD:
                FE07    31  029B    .3          BRW     ILLEGAL                 ; illegal directive
                            029E    .4
                            029E    .5 STARTOK:
                5A  56  C1  029E    .6          ADDL3   R6,R10,FIELDEND(FP)     ; compute and save ending address
                  FC AD     02A1
-3              59  56  D1  02A3   972          CMPL    R6,R9                   ; was that much space remaining?
                    CC  14  02A6   973          BGTR    INSERT_OVF              ; no, take error here
                        05  02A8   974          RSB                             ; return
                            02A9   975
                            02A9   976 ;
                            02A9   977 ; Set up registers so that if fill is needed, a phony call is made
                            02A9   978 ; to REPEATIT with the length in R6 and the 'blank' character in R4
                            02A9   979 ;
                            02A9   980
                            02A9   981 ENDFIELD:
          54    20  9A  02A9   982          MOVZBL  #^A/ /,R4               ; generate blank fill character
    FC AD       5A  C3  02AC   983          SUBL3   R10,FIELDEND(FP),R6     ; compute remaining field length
                56          02B0
                CD  14  02B1   984          BGTR    REPEATIT                ; if any left, go fill with blanks
    5A    FC AD     D0  02B3   985          MOVL    FIELDEND(FP),R10        ; else truncate by setting back pointer
          59    56  C2  02B7   986          SUBL    R6,R9                   ; subtract negative difference from counter
                        05  02BA   987          RSB                             ; return
```

```
-1                                      .SBTTL  PERCENT - Time directives and plural 'S'

                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       These directives are for date and time conversion, and for
                                ;       conditionally inserting a plural 'S' into messages.
                                ;       The time directives insert an ASCII time string into the output buffer.
                                ;       The user may supply a quadword binary time to be converted,
                                ;       or have the current date or time inserted.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB/BSB
                                ;
                                ; INPUTS:
                                ;
                                ;       R4  - second character of directive.  D -> convert
                                ;               date and time, T -> convert time only
                                ;               S -> plural indicator
                                ;       R9  - remaining length of output buffer
                                ;       R10 - current output buffer position
                                ;       R11 - next parameter address
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       none
                                ;
                                ; OUTPUTS:
                                ;
                                ;       none
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       none
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       none
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;--
```

```
-1                                      02BB  1035 PERCENT:
                        03   54   3A     02BB  1036          LOCC    R4,#3,PERCENT_STR       ; find directive type
                 00000027'EF             02BE
                        D6   13          02C3  1037          BEQL    ILLFIELD                ; illegal directive if not found
                        57   D4          02C5  1038          CLRL    R7                      ; assume date and time
                                         02C7  1039          CASE    R0,<10$,30$>,B,#2       ; branch on directive type
                   01'  02   50   8F     02C7                CASEB   R0,#2,S^#<<30007$-30006$>/2>-1
                                         02CB        30006$:
                             0006'       02CB                .SIGNED_WORD    10$-30006$
                             003F'       02CD                .SIGNED_WORD    30$-30006$
                                         02CF        30007$:
                                         02CF  1040
                                         02CF  1041 ;
                                         02CF  1042 ; Time only directive falls through here
                                         02CF  1043 ;
                                         02CF  1044
                        57   D6          02CF  1045          INCL    R7                      ; indicate time only
                                         02D1  1046 10$:                                     ; time and date enters here
                        38   BB          02D1  1047          PUSHR   #^M<R3,R4,R5>           ; save registers
                   20   6A   00   2C     02D3  1048          MOVC5   #0,(R10),#^A/ /,R9,(R10) ; blank fill rest of output buffer
                        6A   59          02D7
                        58   7E   DE     02D9  1049          MOVAL   -(SP),R8                ; space for return length
                        7E   59   7D     02DC  1050          MOVQ    R9,-(SP)                ; form descriptor for output buffer
                        52   6E   DE     02DF  1051          MOVAL   (SP),R2                 ; get address of buffer descriptor
                        51   8B   D0     02E2  1052          MOVL    (R11)+,R1               ; get binary time address
                                         02E5    .1 ;        BEQL    12$                     ; branch if no address                  [01]
                                         02E5    .2 ;        CMPL    (R1),4(R1)              ; let potential access violation        [01]
-2                                       02E5  1055                                          ; ...happen in this frame rather than
                                         02E5  1056                                          ; ...within $ASCTIM to help condition
                                         02E5  1057                                          ; ...handler
                                         02E5  1058 12$:     $ASCTIM_S (R8),(R2),(R1),R7     ; convert time to ascii
                             57   DD     02E5                        PUSHL   R7
                             61   7F     02E7                        PUSHAQ  (R1)
                             62   7F     02E9                        PUSHAQ  (R2)
                             68   3F     02EB                        PUSHAW  (R8)
                             04   FB     02ED                        CALLS   #4,G^SYS$ASCTIM
                      00000000'GF       02EF
                        52   56   D0     02F4  1059          MOVL    R6,R2                   ; did user specify width?
                             03   18     02F7  1060          BGEQ    20$                     ; yes, use it
                        52   68   3C     02F9  1061          MOVZWL  (R8),R2                 ; else use returned length
                                         02FC  1062 20$:
                        59   52   C2     02FC  1063          SUBL    R2,R9                   ; update output length
                             12   19     02FF  1064          BLSS    40$                     ; error, not enough room
                        5A   52   C0     0301  1065          ADDL    R2,R10                  ; update output buffer
                        5E   0C   C0     0304  1066          ADDL    #12,SP                  ; pop locals from stack
                             38   BA     0307  1067          POPR    #^M<R3,R4,R5>           ; restore registers
                                  05     0309  1068          RSB                             ;
                                         030A  1069 30$:
                                         030A  1070
                                         030A  1071 ;
                                         030A  1072 ; Check if the last value converted was equal to one.  If so, then do
                                         030A  1073 ; nothing, else output an 'S' into the output buffer.
                                         030A  1074 ;
                                         030A  1075
                   F8 AD   01   D1       030A  1076          CMPL    #1,LASTVAL(FP)          ; was last value a one
                             13   13     030E  1077          BEQL    60$                     ; yes, simply return
                                03 59   F4   0310  1078          SOBGEQ  R9,50$                  ; check if room in buffer
```

```
                FD94   31  0313  1079 40$:   BRW     OVERFLOW                ; no room , error
                               0316    1
   8A   53 8F   90      0316  1080 50$:   MOVB    #^A/S/,(R10)+           ; plural, insert 'S'
04 FE AA   05   E1      031A  1081         BBC     #5,-2(R10),60$          ; continue if previous character was
                        031F  1082                                         ; ...upper case
   FF AA   20   88      031F  1083         BISB    #^X20,-1(R10)           ; else convert upper 'S' to lower 's'
                        0323    1
                05      0323    2 60$:     RSB                             ; return
```

```
                        0324     .4          .SBTTL  HANDLER - Condition handler
                        0324     .5
-3                      0324  1087 ;++
                        0324  1088 ;
                        0324  1089 ; FUNCTIONAL DESCRIPTION:
                        0324  1090 ;
                        0324  1091 ;       This condition handler is used to catch any errors which
                        0324  1092 ;       ocurred while processing the arguments, such as access
                        0324  1093 ;       violation.  This is because we don't want exceptions
                        0324  1094 ;       occurring within the system service.
                        0324  1095 ;       Care must be taken in this handler to deal with a second access
                        0324  1096 ;       violation while storing the return value for $FAO.
                        0324  1097 ;
                        0324  1098 ; INPUTS:
                        0324  1099 ;
                        0324  1100 ;       CHF$L_SIGARGLST(AP) = Address of signal vector
                        0324  1101 ;       CHF$L_MCHARGLST(AP) = Address of mechanism vector
                        0324  1102 ;
                        0324  1103 ; OUTPUTS:
                        0324  1104 ;
                        0324  1105 ;       The final R0 is set to the status code and the service
                        0324  1106 ;       is exited via $UNWIND.
                        0324  1107 ;--
```

```
                     0324     .2 ;HANDLER:
                     0324     .3 ;        .WORD   0                                                                   [01]
                     0324     .4 ;                                                                                    [01]
                     0324     .5 ;        MOVAB   L^EXE$SIGTORET,(FP)     ;Simple handler for errors here         [01]
                     0324     .6 ;                                                                                    [01]
                     0324     .7 ;        ASSUME  CHF$L_MCHARGLST,EQ,CHF$L_SIGARGLST+4                            [01]
                     0324     .8 ;        MOVQ    CHF$L_SIGARGLST(AP),R0  ; Get address of signal argument list   [01]
                     0324     .9 ;        CMPL    #SS$_UNWIND,CHF$L_SIG_NAME(R0) ;Unwinding?                      [01]
                     0324    .10 ;        BEQL    90$                     ;Exit if yes                            [01]
                     0324    .11 ;        TSTL    CHF$L_MCH_DEPTH(R1)     ;Exception within FAO?                  [01]
                     0324    .12 ;        BNEQ    80$                     ;Resignal if no                         [01]
                     0324    .13 ;        MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;Set final return status [01]
                     0324    .14 ;        CLRQ    -(SP)                   ;Clear depth and new PC arguments       [01]
                     0324    .15 ;        CALLS   #2,G^SYS$UNWIND         ;Unwind to establisher's caller         [01]
                     0324    .16 ;                                        ;***** The next instruction my ACCVIO   [01]
                     0324    .17 ;        MOVL    SF$L_SAVE_AP(FP),R0     ;Get address of FAO's argument list     [01]
                     0324    .18 ;        MOVL    OUTLEN(R0),R0           ;Output length requested?               [01]
                     0324    .19 ;        BEQL    10$                     ;Branch if not                          [01]
                     0324    .20 ;        CLRW    (R0)                    ;Indicate nothing returned in buffer    [01]
                     0324    .21 ;10$:                                    ;***** End of potential ACCVIO          [01]
                     0324    .22 ;80$:    MOVZWL  #SS$_RESIGNAL,R0        ;Resignal (ignore after UNWIND)         [01]
                     0324    .23 ;90$:    RET                             ;                                       [01]
-24                  0324   1132
                     0324   1133         .END
```

```
ARGCOUNT                = 00000000     D
ASC_NAMES                 00000000 R   D  02
CASE_BSB                  00000081 R   D  03
CASE_LOOP                 00000083 R   D  03
CNTRL_LENGTH            = 00000010     D
CNTRL_TABLE               00000010 R   D  02
CR                      = 0000000D     D
CVTASC                    00000104 R   D  03
CVTNUM                    00000178 R   D  03
CVT_BIN_TO_ASC            00000213 R   D  03
DATA_TYPES                00000024 R   D  02
DEF_ARGPTR                00000266 R   D  03
DONE                      000000B3 R   D  03
ENDFIELD                  000002A9 R   D  03
EXCL                    = 00000021     D
EXE$FAO                   00000000 RG  D  03
EXE$FAOL                  00000008 RG  D  03
FAO                       0000000E R   D  03
FAO_CASE                  00000088 R   D  03
FAO_EXIT                  000000B6 R   D  03
FF                      = 0000000C     D
FIELDEND                = FFFFFFFC     D
FIELDS                    0000002A R   D  02
FIRSTARG                = 00000010     D
GETCHAR                   000000C2 P   D  03
GETCOUNT                  000000CF R   D  03
ILLEGAL                   000000A5 R   D  03
ILLFIELD                  0000029B R   D  03
INCR_ARGPTR               00000263 R   D  03
INDSC                   = 00000004     D
INLEN                   = FFFFFFF0     D
INPTR                   = FFFFFFF4     D
INSERT_CHAR               00000271 R   D  03
INSERT_IT                 00000277 R   D  03
INSERT_OVF                00000274 R   D  03
LASTVAL                 = FFFFFFF8     D
LF                      = 0000000A     D
MAIN_SCAN                 0000001B R   D  03
NEWLINE                   00000269 R   D  03
OCT_HEX_DIGITS            00000C31 R   D  02
ONE_CHAR_INDEX          = 00000008     D
ONE_CHAR_CNTRLS           00000018 R   D  02
OUTDSC                  = 0000000C     D
OUTLEN                  = 00000008     D
OVERFLOW                  000000AA R   D  03
PARSE_DIRECTIVE           00000046 R   D  03
PERCENT                   000002BB R   D  03
PERCENT_STR               00000C27 R   D  02
RADIX                     00000038 R   D  02
REPEATIT                  00000280 R   D  03
REPLACEMENT               00000C2D R   D  02
REPLAC_CHRS               00000C1C R   D  02
REPL_OFFSET             = 0000000C     D
SS$_BADPARAM            = 00000014     D
SS$_BUFFEROVF           = 00000601     D
SS$_NORMAL              = 00000001     D
STARTFIELD                00000297 R   D  03
STARTOK                   0000029E R   D  03
STRING_TYPES              00000020 R   D  02
SYS$ASCTIM                ********  GX   03
TAB                     = 00000009     D
TWO_CHAR_CNTRLS           00000010 R   D  02
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	0000003D (61.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000324 (804.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ARGCOUNT	=00000000	83 (1)	
ASC_NAMES	00000000-R	106 (1)	#-808 (1)
CASE_BSB	00000081-R	332 (1)	#-335 (1)
CASE_LOOP	00000083-R	334 (1)	#-331 (1)
CNTRL_LENGTH	=00000010	143 (1)	145 (1) #-313 (1) #-324 (1)
CNTRL_TABLE	00000010-R	123 (1)	143 (1) 145 (1) 147 (1) 313 (1)
CR	=0000000D	94 (1)	#-922 (1)
CVTASC	00000104-R	545.3 (1)	361 (1)
CVTNUM	00000178-R	706 (1)	#-363 (1)
CVT_BIN_TO_ASC	00000213-R	795 (1)	
DATA_TYPES	00000024-R	151 (1)	710 (1)
DECR_ARGPTR	00000266-R	904 (1)	361 (1)
DONE	000000B3-R	372.2 (1)	#-284 (1)
ENDFIELD	000002A9-R	981 (1)	361 (1)
EXCL	=00000021	96 (1)	158 (1) #-275 (1)
EXE$FAO	00000000-R	246 (1)	
EXE$FAOL	00000008-R	257 (1)	
FAO	0000000E-R	262 (1)	#-251 (1)
FAO_CASE	00000088-R	348 (1)	#-333 (1)
FAO_EXIT	000000B6-R	375 (1)	#-366.2 (1) #-372 (1)
FF	=0000000C	98 (1)	158 (1)
FIELDEND	=FFFFFFFC	92 (1)	#-968.6 (1) #-983 (1) #-985 (1)
FIELDS	0000002A-R	155 (1)	#-716 (1) #-756 (1)
FIRSTARG	=00000010	87 (1)	250 (1) #-259.2 (1)
GETCHAR	000000C2-R	422 (1)	#-287 (1) #-302 (1) #-311 (1) #-317 (1)
			#-321 (1) #-486 (1) #-497 (1)
GETCOUNT	000000CF-R	472 (1)	#-301 (1) #-310 (1)
ILLEGAL	000000A5-R	366 (1)	#-314 (1) #-424 (1) #-637 (1) #-712 (1)
			#-968.3 (1)
ILLFIELD	0000029B-R	968.2 (1)	#-1037 (1) #-951 (1)
INCR_ARGPTR	00000263-R	892.3 (1)	361 (1)
INDSC	=00000004	84 (1)	#-264 (1)
INLEN	=FFFFFFF0	89 (1)	#-275 (1) #-279 (1) #-280 (1) #-423 (1)
INPTR	=FFFFFFF4	90 (1)	275 (1) 283 (1) #-285 (1) #-425 (1)
			#-426 (1) #-474 (1) #-478 (1) #-480 (1)
			#-489 (1)
INSERT_CHAR	00000271-R	925 (1)	361 (1)
INSERT_IT	00000277-R	931.5 (1)	#-931 (1)
INSERT_OVF	00000274-R	931.2 (1)	#-833 (1) #-920 (1) #-953 (1) #-973 (1)
LASTVAL	=FFFFFFF8	91 (1)	#-1076 (1) #-788 (1)
LF	=0000000A	95 (1)	158 (1)
MAIN_SCAN	0000001B-R	273 (1)	#-336 (1)
NEWLINE	00000269-R	913 (1)	361 (1)
OCT_HEX_DIGITS	00000031-R	167 (1)	#-733 (1)
ONECHAR_INDEX	=00000008	145 (1)	#-315 (1)
ONE_CHAR_CNTRLS	00000018-R	133 (1)	145 (1)
OUTDSC	=0000000C	86 (1)	#-265 (1) #-378 (1)
OUTLEN	=00000008	85 (1)	#-376 (1) #-378 (1)
OVERFLOW	000000AA-R	369 (1)	#-1079 (1) #-282 (1) #-635 (1) #-931.3 (1)
PARSE_DIRECTIVE	00000046-R	297 (1)	

```
PERCENT                         000002BB-R    1035    (1)       361    (1)
PERCENT_STR                     00000027-R    153     (1)       1036   (1)
RADIX                           00000038-R    171     (1)    #-719     (1)
REPEATIT                        00000280-R    948     (1)       361    (1)     #-984     (1)
REPLACEMENT                     0000002D-R    157     (1)    #-940     (1)
REPLACE_CHRS                    0000001C-R    138     (1)       147    (1)
REPL_OFFSET                    =0000000C      147     (1)    #-940     (1)
SS$_BADPARAM                   =00000014                     #-366.1   (1)
SS$_BUFFEROVF                  =00000601                     #-370     (1)
SS$_NORMAL                     =00000001                     #-372.3   (1)
STARTFIELD                      00000297-R    966     (1)       361    (1)
STARTOK                         0000029E-R    968.5   (1)    #-968     (1)
STRING_TYPES                    00000020-R    149     (1)       552    (1)
SYS$ASCTIM                      00000000-XR                     1058   (1)
TAB                            =00000009      97      (1)       158    (1)
TWO_CHAR_CNTRLS                 00000010-R    124     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...							
$ASCTIM_S	1	1058 (1)	1058 (1)							
$DEFINI	1	75 (1)	75 (1)							
$PUSHADR	1	1058 (1)	1058 (1)							
$SSDEF	21	75 (1)	75 (1)							
CASE	1	349 (1)	1039 (1)	349	(1)	558	(1)	726	(1)	

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.10	00:00:00.23
Command processing	140	00:00:00.69	00:00:01.83
Pass 1	530	00:00:08.56	00:00:16.22
Symbol table sort	0	00:00:00.73	00:00:00.83
Pass 2	242	00:00:04.51	00:00:12.44
Symbol table output	8	00:00:00.07	00:00:00.07
Psect synopsis output	4	00:00:00.03	00:00:00.03
Cross-reference output	23	00:00:00.30	00:00:00.30
Assembler run totals	985	00:00:15.01	00:00:31.95

The working set limit was 1000 pages.
36659 bytes (72 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 461 non-local and 55 local symbols.
1158 source lines were read in Pass 1, producing 0 object records in Pass 2.
12 pages of virtual memory were used to define 10 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	7

487 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) FAO/UPDA=(FAO.UPD,FAO.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DIAG/
```

Table of contents

(1)	71	DECLARATIONS
(2)	196	FIL$OPENFILE - RETURN FILE HEADER AND STATISTICS BLOCK
(3)	467	FIL$CACHE_INIT - INIT FILEREAD CACHE
(4)	555	FIL$CACHE_TRUNC - TRUNCATE FILEREAD CACHE
(5)	605	ASSIGN_DEV - ASSIGN A DEVICE AND RETURN A CHANNEL
(6)	649	STORE3DIGITS - STORE 3 ASCII DIGITS
(7)	684	FORMDIRSTRING - GET A DIRECTORY STRING
(8)	751	MOUNT - MOUNT THE VOLUME, INIT FOR FILE LOOKUP
(9)	867	FINDFILID - FIND FILE ID FOR SPECIFIED FILE
(10)	1134	FIL$FINDFILID - STRUCTURE LEVEL 2
(11)	1255	FIL$FINDFILID - STRUCTURE LEVEL 1
(12)	1338	READ_DIR_LBN - READ NEXT DIRECTORY LBN
(13)	1381	RDCHKFILHDR - READ AND CHECK FILE HEADER
(14)	1531	READVBN, WRITEVBN - READ/WRITE VIRTUAL BLOCK
(15)	1627	INIRTRVPTRSCAN - INITIALIZE RETRIEVAL POINTER SCAN
(16)	1660	GETRTRVPTR - CONVERT NEXT RETRIEVAL POINTER
(17)	1757	STATBLK - GET FILE STATISTICS BLOCK
(18)	1874	FIL$CHKFILHDR - CHECK FILE HEADER VALIDITY
(19)	1946	CHECKSUM - VALIDATE A CHECKSUM

```
                        0000     1          .TITLE  FILEREAD - FILES11 LEVEL 1 & 2 FILE READING ROUTINES
                        0000    .1          .IDENT  '06-02'
-1                      0000     3
                        0000     4  ;
                        0000     5  ;*****************************************************************************
                        0000     6  ;*                                                                           *
                        0000    .1  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983 BY                                  *
-1                      0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
                        0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
                        0000    10  ;*                                                                           *
                        0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                        0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
                        0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
                        0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
                        0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
                        0000    16  ;*  TRANSFERRED.                                                             *
                        0000    17  ;*                                                                           *
                        0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
                        0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
                        0000    20  ;*  CORPORATION.                                                             *
                        0000    21  ;*                                                                           *
                        0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
                        0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000    24  ;*                                                                           *
                        0000    25  ;*                                                                           *
                        0000    26  ;*****************************************************************************
                        0000    27  ;
                        0000    28  ;++
                        0000    29  ; FACILITY:     USER CALLABLE PROCEDURES
                        0000    30  ;
                        0000    31  ; ABSTRACT:
                        0000    32  ;
                        0000    33  ;       THIS SET OF ROUTINES PROVIDES THE CAPABILITY OF ''OPENING'' AND
                        0000    34  ; READING FILES BY FILE NAME FROM A FILES11 STRUCTURE LEVEL 1 OR 2 VOLUME.
                        0000    35  ; THERE IS NO MULTI-VOLUME SUPPORT, AND MULTI-HEADER SUPPORT IS LIMITED
                        0000    36  ; TO RETURNING THE CORRECT FILE SIZE IN THE STATBLK.
                        0000    37  ;
                        0000    38  ; ENVIRONMENT:  USER MODE
                        0000    39  ;
                        0000    40  ; AUTHOR: PETER H. LIPMAN       , CREATION DATE: 14-DEC-76
                        0000    41  ;
                        0000    42  ; MODIFIED BY:
                        0000    .1  ;       02      Bob Bergazzi    Jun 29,1983     Version 6.12
                        0000    .2  ;               Adapted for supervisor 6.12.
                        0000    .3  ;
                        0000    43  ;       V03-001 PHL0103         Peter H. Lipman         20-Jul-1982
                        0000    44  ;               Correct error in FIL$FINDFILID for structure level 1
                        0000    45  ;               when directory is cached.
                        0000    46  ;
                        0000    47  ;       V02-007 PHL0019         Peter H. Lipman         31-Oct-1981
                        0000    48  ;               Fix bug in FIL$RDCHKFILHDR having to do with overflowing
                        0000    49  ;               the callers retrieval pointer buffer for multi-header files.
                        0000    50  ;               Fix documentation for FIL$STATBLK showing that the returned
                        0000    51  ;               retrieval pointer length is the number of bytes that would
                        0000    52  ;               have been stored if the buffer had been large enough.
                        0000    53  ;
                        0000    54  ;       V02-006 PHL0009         Peter H. Lipman         06-May-1981
```

```
0000     55 ;                  TOPSYS directory was not being used properly.
0000     56 ;
0000     57 ;       V02-005 PHL0007         Peter H. Lipman         14-Mar-1981
0000     58 ;               Add cacheing of directory lookups, directory header info,
0000     59 ;               directory data blocks, and index file header.
0000     60 ;               Change interface to FIL$RDWRTLBN to allow the reading
0000     61 ;               of multiple blocks.
0000     62 ;               Implement multiple level directory lookups and a new
0000     63 ;               top level directory in which all the system directories
0000     64 ;               are found.
0000     65 ;       V02-004 PHL0006         Peter H. Lipman         13-Dec-1980
0000     66 ;               Add capability to return retrieval pointer information
0000     67 ;               and thus allow all non-contiguous files to be handled
0000     68 ;               at boot time.
0000     69 ;--
```

```
                     0000     71          .SBTTL  DECLARATIONS
                     0000     72 ;
                     0000     73 ; INCLUDE FILES:
                     0000     74 ;
                     0000     75          .nocross
                     0000     76          $DIRDEF                         ;DIRECTORY ENTRY OFFSET DEFINITIONS
                     0000     77          $FATDEF                         ;RECORD ATTRIBUTE AREA DEFINITIONS
                     0000     78          $FH1DEF                         ;FILE HEADER DEFINITIONS, LEVEL 1
                     0000     79          $FH2DEF                         ;FILE HEADER DEFINITIONS, LEVEL 2
                     0000     80          $FM1DEF                         ;       MAP AREA, LEVEL 1
                     0000     81          $FM2DEF                         ;       MAP AREA, LEVEL 2
                     0000     82          $FIDDEF                         ;FILE ID OFFSET DEFINITIONS
                     0000     83          $HM1DEF                         ;HOME BLOCK DEFINITIONS, LEVEL 1
                     0000     84          $HM2DEF                         ;HOME BLOCK DEFINITIONS, LEVEL 2
                     0000     85          $IODEF                          ;I/O DEFINITIONS
                     0000     86          $PSLDEF                         ;PROCESSOR STATUS LONG WORD DEFINITIONS
                     0000     87          $SSDEF                          ;SYSTEM SERVICE DEFINITIONS
                     0000     88          $VADEF                          ;VIRTUAL ADDRESS DEFINITIONS
                     0000     89 ;
                     0000     90 ; MACROS:
                     0000     91 ;
                     0000     92          .MACRO  READVBN CHAN,VBN,BUFADR,HDRADR
                     0000     93          .LIST   MEB
                     0000     94                  PUSHAL  HDRADR
                     0000     95                  PUSHAL  BUFADR
                     0000     96                  PUSHL   VBN
                     0000     97                  PUSHL   CHAN
                     0000     98                  CALLS   #4,W^FIL$READVBN
                     0000     99          .NLIST  MEB
                     0000    100          .ENDM   READVBN
                     0000    101
                     0000    102          .MACRO  READLBN CHAN,VBN,BUFADR
                     0000    103          .LIST   MEB
                     0000    104                  ROTL    #9,#1,-(SP)
                     0000    105                  MOVZWL  #IO$_READLBLK,-(SP)
                     0000    106                  PUSHAL  BUFADR
                     0000    107                  PUSHL   VBN
                     0000    108                  PUSHL   CHAN
                     0000    109                  CALLS   #5,W^FIL$RDWRTLBN
                     0000    110          .NLIST  MEB
                     0000    111          .ENDM   READLBN
                     0000    112          .cross
                     0000    113 ;
                     0000    114 ; EQUATED SYMBOLS:
                     0000    115 ;
                     0000    116          ASSUME  FH1$B_MPOFFSET EQ FH2$B_MPOFFSET
                     0000    117          ASSUME  FH1$W_STRUCLEV EQ FH2$W_STRUCLEV
                     0000    118          ASSUME  FH1$W_CHECKSUM EQ FH2$W_CHECKSUM
                     0000    119          ASSUME  HM1$W_STRUCLEV EQ HM2$W_STRUCLEV
                     0000    120          ASSUME  HM1$W_CHECKSUM1 EQ HM2$W_CHECKSUM1
                     0000    121          ASSUME  HM1$W_CHECKSUM2 EQ HM2$W_CHECKSUM2
                     0000    122          ASSUME  FH1$W_CHECKSUM EQ HM1$W_CHECKSUM2
                     0000    123          ASSUME  FH1$V_CONTIG EQ FH2$V_CONTIG
                     0000    124
000001FE             0000    125          FH1$W_VBNOFFSET = FH1$W_CHECKSUM ;SAVE INDEX FILE VBN OFFSET
                     0000    126                                           ;IN THIS PLACE IN INDEX FILE HEADER
                     0000    127
```

```
                     0000   128          ASSUME  FH1$C_LEVEL1a-8 EQ 1
          00000008   0000   129          FH1$V_LEVEL1 = 8
                     0000   130
                     0000   131          ASSUME  FH2$C_LEVEL2a-8 EQ 2
          00000009   0000   132          FH2$V_LEVEL2 = 9
          0000000A   0000   133          FH2$V_BIGFILNUM = 10            ;IF SET USE HIGH 8 BITS OF FILE ID RVN
                     0000   134                                          ;FIELD AS FILE NUMBER EXTENSION
                     0000   135                                          ;BIT IS PLACED IN FH2$W_STRUCLEV
                     0000   136                                          ;BY THE FIL$MOUNT CODE
                     0000   137
          00000001   0000   138          FIL$C_CACHE_ID = 1              ;VERSION OF THE FILEREAD CACHE
                     0000   139 ;
                     0000   140 ; OFFSETS INTO HEADER PORTION OF THE FILEREAD CACHE
                     0000   141 ;
                     0000   142          $OFFSET 0,POSITIVE,<-
                     0000   143          <FIL$W_CACHE_ID,2>,-            ;CACHE IDENT, AND WRITE INTERLOCK BIT
                     0000   144          <,2>,-                          ;SPARE
                     0000   145          FIL$L_DIROFF,-                  ;OFFSET IN BYTES TO DIRECTORY CACHE
                     0000   146          FIL$L_DIRNXT,-                  ;NEXT OFFSET TO ALLOCATE DIR CACHE ENTRY
                     0000   147          <FIL$L_DIRMAX,0>,-              ;MAX OFFSET FOR DIR CACHE
                     0000   148          FIL$L_LBNOFF,-                  ;OFFSET IN BYTES TO BEGIN OF LBN CACHE
                     0000   149          FIL$L_LBNNXT,-                  ;NEXT OFFSET TO ALLOCATE LBN CACHE
                     0000   150          FIL$L_LBNMAX,-                  ;MAX OFFSET FOR LBN CACHE
                     0000   151          <FIL$A_IXFHDR,512>,-            ;INDEX FILE HEADER
                     0000   152          <FIL$C_SIZE,0>-                 ;START OF DIRECTORY CACHE
                     0000   153          >
                     0000       FIL$W_CACHE_ID:
                     0004       FIL$L_DIROFF:
                     0008       FIL$L_DIRNXT:
                     000C       FIL$L_DIRMAX:
                     000C       FIL$L_LBNOFF:
                     0010       FIL$L_LBNNXT:
                     0014       FIL$L_LBNMAX:
                     0018       FIL$A_IXFHDR:
                     0218       FIL$C_SIZE:
                     0000   154 ;
                     0000   155 ; OFFSETS INTO DIRECTORY CACHE ENTRIES
                     0000   156 ;
                     0000   157          $OFFSET 0,POSITIVE,<-
                     0000   158          <FIL$A_DIR_FID,6>,-             ;DIRECTORY ID
                     0000   159          <FIL$T_DIR_NAM,10>,-            ;COUNTED NAME OF DIRECTORY - NO ".DIR"
                     0000   160          <FIL$Q_DIR_HDR,0>,-             ;DIRECTORY HEADER INFORMATION
                     0000   161          <FIL$W_DIR_BKCNT,2>,-           ;SIZE IN BLOCKS OF DIRECTORY FILE
                     0000   162          <FIL$B_DIR_LVL,1>,-             ;STRUCTURE LEVEL OF DIRECTORY
                     0000   163          <,1>,-                          ;SPARE BYTE
                     0000   164          FIL$L_DIR_LBN,-                 ;STARTING LBN OF DIRECTORY
                     0000   165          FIL$L_DIR_BFOFF,-               ;OFFSET TO DIR LBN BUFFER
                     0000   166          <FIL$W_DIR_BFCNT,2>,-           ;SIZE IN BLOCKS OF DIR LBN BUFFER
                     0000   167          <FIL$A_DIR_OFID,6>,-            ;OUTPUT FILE ID FROM LOOKUP
                     0000   168          <FIL$C_DIR_SIZE,0>-             ;SIZE OF DIRECTORY CACHE ENTRY
                     0000   169          >
                     0000       FIL$A_DIR_FID:
                     0006       FIL$T_DIR_NAM:
                     0010       FIL$Q_DIR_HDR:
                     0010       FIL$W_DIR_BKCNT:
                     0012       FIL$B_DIR_LVL:
                     0014       FIL$L_DIR_LBN:
```

```
                     0018        FIL$L_DIR_BFOFF:
                     001C        FIL$W_DIR_BFCNT:
                     001E        FIL$A_DIR_OFID:
                     0024        FIL$C_DIR_SIZE:
                     0000    170 ;
                     0000    171 ; MAKE THESE GLOBAL SO THAT A CACHE SIZE CAN BE PROPERLY CALCULATED
                     0000    172 ; THE CALCULATION IS:
                     0000    173 ;
                     0000    174 ;       FIL$C_SIZE + (DIRCNT * FIIL$C_DIR_SIZE> + <LBNCNT * 512>
                     0000    175 ;
                     0000    176         .GLOBAL FIL$C_SIZE,FIL$C_DIR_SIZE
                     0000    177 ;
                     0000    178 ; DEFINE THE FOLLOWING WEAK REFERENCES, THEY NEED NOT BE PRESENT
                     0000    179 ;
                     0000    180         .WEAK   FIL$GQ_CACHE            ;DESCRIPTOR FOR FILEREAD CACHE
                     0000    181         .WEAK   FIL$GT_DDDEV            ;ASCIC DEFAULT DEVICE NAME STRING
                     0000    182         .WEAK   FIL$GT_TOPSYS           ;ASCIC TOP LEVEL SYSTEM DIRECTORY
                     0000    183 ;
                     0000    184 ; OWN STORAGE:
                     0000    185 ;
                     0000    186
                00000000      .1        .psect  code, nowrt, exe, rd, shr, long ;                                   [02]
-1                   0000    188
                     0000    189 FIL_GQ_CACHE:
           00000000' 0000    190         .ADDRESS FIL$GQ_CACHE
                     0004    191 FIL_GT_DDDEV:
           00000000' 0004    192         .ADDRESS FIL$GT_DDDEV
                     0008    193 FIL_GT_TOPSYS:
           00000000' 0008    194         .ADDRESS FIL$GT_TOPSYS
```

```
                                000C   196          .SBTTL  FIL$OPENFILE - RETURN FILE HEADER AND STATISTICS BLOCK
                                000C   197 ;++
                                000C   198 ; FUNCTIONAL DESCRIPTION:
                                000C   199 ;
                                000C   200 ;       THE OPENFILE ROUTINE ACCEPTS A FULL FILE NAME IN THE FORMAT
                                000C   201 ;       DEV:[DIR]FILE.TYP;VERSION.
                                000C   202 ; IT ASSIGNS AND RETURNS A CHANNEL, READS THE FILE HEADER, RETURNS THE
                                000C   203 ; STATISTICS BLOCK, AND OPTIONALLY RETURNS THE RETRIEVAL POINTERS IN
                                000C   204 ; A NORMALIZED (LONG WORD COUNT, LONG WORD LBN) FORMAT.
                                000C   205 ; THE DIRECTORY MAY BE IN ANY OF THE STANDARD FORMATS:
                                000C   206 ;       [10,40], [010040], [ABCDEFGHI], OR WITH < AND > REPLACING [ AND ].
                                000C   207 ; VERSION MAY BE ZERO IN WHICH CASE THE HIGHEST VERSION IS FOUND
                                000C   208 ;
                                000C   209 ; CALLING SEQUENCE:
                                000C   210 ;
                                000C   211 ;       CALLG   ARGLIST,FIL$OPENFILE
                                000C   212 ;
                                000C   213 ; INPUT PARAMETERS:
                                000C   214 ;
                                000C    .1 ;       CHANADR(AP)     =       ADDRESS OF RPB                                  [02]
-1                              000C   216 ;       FILNAM(AP)              ADDRESS OF 2 LONG WORD FILE NAME STRING DESCRIPTOR
                                000C   217 ;                               1 - SIZE OF STRING
                                000C   218 ;                               2 - ADDRESS OF STRING
                                000C   219 ;                                       DB1:[10,40]FILTST.EXE
                                000C   220 ;       IXFHDR(AP)              ADDRESS OF 512 BYTE BUFFER TO BE USED FOR
                                000C   221 ;                                       THE INDEX FILE HEADER
                                000C   222 ;       FILHDR(AP)              ADDRESS OF 512 BYTE BUFFER TO RETURN FILE HEADER
                                000C   223 ;       STATBLK(AP)             ADDRESS OF 2 LONG WORD BLOCK IN WHICH THE
                                000C   224 ;                               FOLLOWING WILL BE RETURNED
                                000C   225 ;                               1 - LOGICAL BLOCK NUMBER OF FIRST BLOCK OF
                                000C   226 ;                                       FILE OR 0 IF FILE IS NOT CONTIGUOUS
                                000C   227 ;                               2 - SIZE OF FILE IN BLOCKS
                                000C   228 ;       RTRVPTRLEN(AP)  =       ADDRESS TO RETURN THE NUMBER OF
                                000C   229 ;                               BYTES OF RETRIEVAL POINTERS STORED
                                000C   230 ;                               ***** OPTIONAL PARAMETER *****
                                000C   231 ;       RTRVPTRBUF(AP)  =       ADDRESS OF RETRIEVAL POINTER
                                000C   232 ;                               BUFFER DESCRIPTOR.  THIS PARAMETER
                                000C   233 ;                               IS PRESENT IF AND ONLY IF
                                000C   234 ;                               RTRVPTRLEN IS PRESENT.
                                000C   235 ;                               THE RETRIEVAL POINTERS ARE RETURNED IN
                                000C   236 ;                               THE FORM 32 BIT BLOCK COUNT, 32 BIT LBN
                                000C   237 ;                               A ZERO BUFFER DESCRIPTOR ADDRESS OR A
                                000C   238 ;                               ZERO BUFFER ADDRESS MEANS DON'T
                                000C   239 ;                               RETURN RETRIEVAL POINTER INFO
                                000C   240 ;
                                000C   241 ; IMPLICIT INPUTS:
                                000C   242 ;
                                000C   243 ;       NONE
                                000C   244 ;
                                000C   245 ; OUTPUT PARAMETERS:
                                000C   246 ;
                                000C   247 ;       R0 = SYSTEM STATUS CODE
                                000C   248 ;
                                000C   249 ; IMPLICIT OUTPUTS:
                                000C   250 ;
                                000C   251 ;       NONE
                                000C   252 ;
```

```
                                000C   253 ; COMPLETION CODES:
                                000C   254 ;
                                000C   255 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                                000C   256 ;       SS$_NOSUCHFILE          FAILED TO FIND DIRECTORY OR FILE
                                000C   257 ;       SS$_BADFILENAME         SYNTAX ERROR IN DIRECTORY OR FILE NAME STRING
                                000C   258 ;
                                000C   259 ; THE FOLLOWING COMPLETION CODES INDICATE FILE STRUCTURE PROBLEMS
                                000C   260 ;
                                000C   261 ;       SS$_BADCHKSUM           CHECKSUM ERROR IN HOME BLOCK, INDEX FILE HEADER
                                000C   262 ;                               DIRECTORY FILE HEADER OR FILE HEADER
                                000C   263 ;       SS$_BADFILEHDR          FILE HEADER CONSISTENCY CHECK FAILED FOR
                                000C   264 ;                               INDEX FILE, DIRECTORY FILE, OR DESIRED FILE
                                000C   265 ;       SS$_FILESTRUCT          HOME BLOCK INDICATES THAT THIS VOLUME
                                000C   266 ;                               CONTAINS A NON-SUPPORTED FILE STRUCTURE
                                000C   267 ;                               OR POSSIBLY THE HOMEBLOCK IS GARBAGE
                                000C   268 ;
                                000C   269 ; SIDE EFFECTS:
                                000C   270 ;
                                000C   271 ;       NONE
                                000C   272 ;
                                000C   273 ; EQUATED SYMBOLS
                                000C   274 ;
                                000C   275 ;    OFFSETS FROM AP
                                000C   276 ;
00000000                        000C   277          ARGCNT          = 0
00000004                        000C   278          CHANADR         = 4
00000008                        000C   279          FILNAM          = 8
0000000C                        000C   280          IXFHDR          = 12
00000010                        000C   281          FILHDR          = 16
00000014                        000C   282          STATBLK         = 20
00000018                        000C   283          RTRVPTRLEN      = 24            ; OPTIONAL
0000001C                        000C   284          RTRVPTRBUF      = 28            ; PRESENT IF AND ONLY IF RTRVPTRLEN IS
                                000C   285 ;
                                000C   286 ;    OFFSETS FROM FP
                                000C   287 ;
                                000C   288          $OFFSET 0,NEGATIVE,<-
                                000C   289          <FID,6>,-                      ;3 WORD FILE IDENTIFIER
                                000C    .1          <DIRNAM,46>,-                  ;DIRECTORY NAME AREA
                                000C    .2          <NAMBLK,40>,-                  ;5 WORD NAME BLOCK AREA
-2                              000C   292          <NAMDSC,12>,-                  ;NAME DESCRIPTOR AREA
                                000C   293          <SCRATCHSIZE,0>-               ;SIZE OF SCRATCH AREA
                                000C   294          >
                                FFFA        FID:
                                FFCC        DIRNAM:
                                FFA4        NAMBLK:
                                FF98        NAMDSC:
                                FF98        SCRATCHSIZE:
                                000C   295 ;
                                000C   296 ; THE FILE DESCRIPTION ON THE STACK LOOKS AS FOLLOWS:
                                000C   297 ;
                                000C   298 ;       +-----------------------------+
                                000C   299 ;       |   DIRECTORY NAME COUNT      |     :NAMDSC
                                000C   300 ;       |-----------------------------|
                                000C   301 ;       | ADDRESS OF DIRECTORY NAME   |
                                000C   302 ;       +-----------------------------+
                                000C   303 ;       |      ADDRESS OF NAMBLK      |
                                000C   304 ;       +-----------------------------+
```

```
                                000C   305 ;       |                           |   :NAMBLK
                                000C   306 ;       +---------------------------+
                                000C   307 ;       |                           |
                                000C   308 ;       +---------------------------+
                                000C   309 ;       |                           |
                                000C   310 ;       |                           |
                                000C   311 ;       +---------------------------+
                                000C   312 ;       |                           |   :DIRNAM
                                000C   313 ;       |     ASCII STRING OF       |
                                000C   314 ;       |      DIRECTORY FILE       |
                                000C   315 ;       |     NAME.TYPE;VERSION     |
                                000C   316 ;       |                           |
                                000C   317 ;       +---------------------------+
                                000C   318 ;       |         FILE ID           |   :FILID
                                C00C   319 ;       |         NUMBER            |
                                000C   320 ;       +---------------------------+
                                000C   321 ;       |  RELATIVE VOLUME NUMBER   |
                                000C   322 ;       +---------------------------+
                                000C   323 ;
                                000C   324 ;--
                                000C   325
                                000C   326          .ENABL  LSB
                                000C   327
                                000C   328 FIL$OPENFILE::
                                000C    .1 ;        .WORD   ^M<R2,R3,R4,R5,R6,R7,R11>                                   [02]
                                000C    .2 ;        BSBW    ASSIGN_DEV              ;ASSIGN THE DEVICE ONCE IN CALLER'S MODE
                                000C    .3 ;        BLBS    R0,OPENFILE_2           ;BRANCH IF SUCCESSFUL               [02]
                                000C    .4 ;        RET                                                                 [02]
-4                              000C   333 OPENFILE_1:
                          08FC  000C   334          .WORD   ^M<R2,R3,R4,R5,R6,R7,R11>
                                000E   335 OPENFILE_2:
                 00000068 8F    C2  000E   336          SUBL    #-SCRATCHSIZE,SP        ;RESERVE SCRATCH STORAGE
                             5E        0014
               A0 AD    A4 AD    DE  0015   337          MOVAL   NAMBLK(FP),NAMDSC+8(FP) ;SET ADDRESS OF NAME BLOCK
                                001A   338 ;
                                001A   339 ; IF CACHE DESCRIPTOR EXISTS AND IS IN SYSTEM SPACE, THEN WE
                                001A   340 ; HAD BETTER BE IN KERNEL MODE TO USE THE CACHE.
                                001A   341 ;
                5B    FFE2 CF    D0  001A   342          MOVL    W^FIL_GQ_CACHE,R11      ;IS CACHE IN SYSTEM SPACE?
                           02    14  001F   343          BGTR    10$                     ;BRANCH IF DESCRIPTOR PRESENT
                                0021   344                                             ;AND NOT IN SYSTEM SPACE
                           10    13  0021   345          BEQL    20$                     ;BRANCH IF NO DESCRIPTOR PRESENT
                                0023    .1 ;        MOVPSL  R0                      ;GET PSL                            [02]
                                0023    .2 ;        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R0,R0 ;FETCH CURRENT MODE       [02]
                                0023    .3 ;        BEQL    10$                     ;BRANCH IF ALREADY IN KERNEL MODE   [02]
                                0023    .4 ;        $CMKRNL_S B^OPENFILE_1,(AP)     ;CALL THIS PROCEDURE IN KERNEL MODE [02]
                                0023    .5 ;        CMPL    R0,#SS$_NOPRIV          ;ASSUME NOPRIV MEANS WE COULDN'T    [02]
                                0023    .6 ;                                        ;GET IN TO KERNEL MODE              [02]
                                0023    .7 ;        BEQL    15$                                                         [02]
                                0023    .8 ;        RET                                                                 [02]
-8           00000004'EF    D0  0023   354 10$:     MOVL    FIL$GQ_CACHE+4,R11      ;IS THE CACHE ENABLED?
                             5B        0029
                           07    13  002A   355          BEQL    20$                     ;BRANCH IF NOT
                        01 6B    B1  002C   356          CMPW    FIL$W_CACHE_ID(R11),#FIL$C_CACHE_ID ;CORRECT VERSION OF CACHE?
                           02    13  002F   357          BEQL    20$                     ;BRANCH IF YES
                           5B    D4  0031   358 15$:     CLRL    R11                     ;DISABLE THE CACHE
                 00000000'EF    DE  0033   359 20$:     MOVAL   FIL$GT_DDSTRING,R7      ;ADDRESS OF COUNTED STRING
```

```
                57          0039
          56    87    9A    003A    360          MOVZBL  (R7)+,R6                ;GET BYTE COUNT
                57    D6    003D    361          INCL    R7                      ;STEP OVER BRACKET
          56    02    C2    003F    362          SUBL    #2,R6                   ;DON'T COUNT THE BRACKETS
                            0042    363 ;
                            0042    364 ; GET FILE NAME STRING, AND STRIP DEVICE OFF IF PRESENT
                            0042    365 ;
     50   08 AC    D0    0042    366          MOVL    FILNAM(AP),R0           ;ADDRESS OF FILE NAME DESCRIPTOR
                27    13    0046    367          BEQL    32$                     ;BRANCH IF NO NAME SPECIFIED
          52    60    7D    0048    368          MOVQ    (R0),R2                 ;R2 = SIZE, R3 = ADDRESS
     63   52    3A    3A    004B    369          LOCC    #^A/:/,R2,(R3)          ;DEVICE NAME PRESENT?
                07    13    004F    370          BEQL    25$                     ;BRANCH IF NOT
     53   01 A1    9E    0051    371          MOVAB   1(R1),R3                ;ADDRESS BEYOND ":"
          52    70    9E    0055    372          MOVAB   -(R0),R2                ;REMAINING SIZE
                            0058    373 ;
                            0058    374 ; SEE IF DIRECTORY SPECIFIED IN THE FILE NAME STRING
                            0058    375 ;
     63   5B 8F    91    0058    376 25$:     CMPB    #^A/[/,(R3)             ;DIRECTORY DELIMITER?
                05    13    005C    377          BEQL    30$                     ;BRANCH IF YES
          63    3C    91    005E    378          CMPB    #^A/</,(R3)             ;ALTERNATE CHARACTER
                1D    12    0061    379          BNEQ    40$                     ;BRANCH IF NO DIRECTORY SPECIFIED
     50   33    02    81    0063    380 30$:     ADDB3   #2,(R3)+,R0             ;SCAN FOR MATCHING BRACKEY ] OR >
                52    D7    0067    381          DECL    R2                      ;ADJUST SIZE AND ADR OF STRING
     63   52    50    3A    0069    382          LOCC    R0,R2,(R3)              ;SCAN FOR CLOSE BRACKET
                03    12    006D    383          BNEQ    35$                     ;BRANCH IF FOUND IT
              03E7    31    006F    384 32$:     BRW     BADFILNAM               ;BAD FILE NAME IF NO CLOSE BRACKET
          57    53    D0    0072    385 35$:     MOVL    R3,R7                   ;ADDRESS OF DIRECTORY NAME
     56   51    53    C3    0075    386          SUBL3   R3,R1,R6                ;SIZE OF DIRECTORY NAME
          52    70    9E    0079    387          MOVAB   -(R0),R2                ;SIZE REMAINING SKIP CLOSE BRACKET
     53   01 A1    DE    007C    388          MOVAL   1(R1),R3                ;ADR OF REMAINING STRING BEYOND CLOSE BRACKE
                            0080    389 ;
                            0080    390 ; SET UP COMMON ARGUMENT LIST FOR MOUNT, FINDFILID, RDCHKFILHDR
                            0080    391 ;
                            0080    392 ;          +-------------------------+
                            0080    393 ;          |      ARGUMENT COUNT     |    : AP
                            0080    394 ;          +-------------------------+
                            0080    395 ;          |      CHANNEL NUMBER     |
                            0080    396 ;          +-------------------------+
                            0080    397 ;          |     NAME DESCRIPTOR     |
                            0080    398 ;          +-------------------------+
                            0080    399 ;          | INDEX FILE HEADER BUF ADR |
                            0080    400 ;          +-------------------------+
                            0080    401 ;          |  FILE HEADER BUFFER ADDR  |
                            0080    402 ;          +-------------------------+
                            0080    403 ;          | ADDR OF STATISTICS BLOCK  |
                            0080    404 ;          +-------------------------+
                            0080    405 ;          | ADDRESS OF FILE ID BLOCK  |
                            0080    406 ;          +-------------------------+
                            0080    407 ;          |  ADDR OF RTRV PTR LENGTH  |
                            0080    408 ;          +-------------------------+
                            0080    409 ;          | ADDR OF RTRV PTR BUF DSCR |
                            0080    410 ;          +-------------------------+
                            0080    411 ;
                7E    7C    0080    412 40$:     CLRQ    -(SP)                   ;ASSUME NO RETRIEVAL POINTERS REQUESTED
          07    6C    D1    0082    413          CMPL    ARGCNT(AP),#RTRVPTRBUF/4 ;RETRIEVAL POINTER PARAMETERS PRESENT?
                04    19    0085    414          BLSS    45$                     ;BRANCH IF NOT
     6E   18 AC    7D    0087    415          MOVQ    RTRVPTRLEN(AP),(SP)     ;PUT RTRV PTR PARAMS IN LIST
```

```
                FA AD    DF  008B   416 45$:    PUSHAL  FID(FP)                         ;ADDRESS OF FILE ID
             7E 10 AC    7D  008E   417          MOVQ    FILHDR(AP),-(SP)            ;PUSH STATBLK ADR, FILHDR ADR
                0C AC    DD  0092   418          PUSHL   IXFHDR(AP)                  ;INDEX FILE HEADER ADDRESS
                98 AD    DF  0095   419          PUSHAL  NAMDSC(FP)                  ;ADR OF 3 LONG WORD NAME DESCRIPTOR
                04 AC    DD  0098    .1          PUSHL   CHANADR(AP)                 ;CHANNEL TO USE, LONG WORD FOR BOOTING  [02]
-1                 06    DD  009B   421          PUSHL   #6                          ;PARAMETER COUNT
                   5B    D5  009D   422          TSTL    R11                         ;CACHE ENABLED?
                   07    13  009F   423          BEQL    50$
             0C AE 18 AB    DE  00A1   424          MOVAL   FIL$A_IXFHDR(R11),IXFHDR(SP) ;USE CACHED INDEX FILE HEADER
                   08    11  00A6   425          BRB     60$                         ;AND SKIP THE MOUNT
          025D'CF  6E    FA  00A8   426 50$:     CALLG   (SP),W^FIL$MOUNT            ;'MOUNT THE VOLUME'' (READ HOME
                         00AD   427                                                  ;BLOCK, INDEX FILE HEADER, GET
                         00AD   428                                                  ;STRUCTURE LEVEL OF VOLUME)
                61 50    E9  00AD   429          BLBC    R0,100$                     ;BRANCH IF ERROR
                         00B0   430 ;
                         00B0   431 ; SET UP FOR THE DIRECTORY LOOK UP
                         00B0   432 ;
             FA AD   04  B0  00B0   433 60$:     MOVW    #FID$C_MFD,FID(FP)          ;MFD FILE NUMBER
             FC AD   04  D0  00B4   434          MOVL    #FID$C_MFD,FID+2(FP)        ;MFD FILE SEQUENCE NO., RVN = 0
                FF4C CF  D5  00B8   435          TSTL    W^FIL_GT_TOPSYS             ;TOP LEVEL SYSTEM DIRECTORY PRESENT?
                   1A    13  00BC   436          BEQL    70$                         ;BRANCH IF NOT
          00000000'EF    DE  00BE   437          MOVAL   FIL$GT_TOPSYS,R1            ;GET ADDRESS OF TOP LEVEL DIR STRING
                   51        00C4
             50    81    9A  00C5   438          MOVZBL  (R1)+,R0                    ;GET SIZE TO R0, ADR TO R1
                   0E    13  00C8   439          BEQL    70$                         ;BRANCH IF NONE SPECIFIED
             7E    56    7D  00CA   440          MOVQ    R6,-(SP)                    ;SAVE DIRECTORY STRING DESCRIPTOR
             56    50    7D  00CD   441          MOVQ    R0,R6                       ;TREAT TOPSYS LIKE DIR STRING
                011E    30  00D0   442          BSBW    FORMDIRSTRING               ;FORM THE DIRECTORY NAME
             56    8E    7D  00D3   443          MOVQ    (SP)+,R6                    ;RESTORE READ DIRECTORY DESCRIPTOR
                   03    11  00D6   444          BRB     75$
                0116    30  00D8   445 70$:     BSBW    FORMDIRSTRING               ;GET NEXT DIRECTORY TO LOOKUP
          98 AD    50    7D  00DB   446 75$:     MOVQ    R0,NAMDSC(FP)               ;STORE DESCRIPTOR OF ITS NAME
                   6E    DF  00DF   447 80$:     PUSHAL  (SP)                        ;REAL ADDRESS OF ARGUMENT LIST
                   5B    DD  00E1   448          PUSHL   R11                         ;CACHE ADDRESS IF ANY
          02ED'CF  02    FB  00E3   449          CALLS   #2,W^FIL$FINDFILID          ;FIND THE FILE ID
                26 50    E9  00E8   450          BLBC    R0,100$                     ;BRANCH IF ERROR
                   3F    BB  00EB    .1         PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; SAVE THE REGISTER
             00 8F   28    28  00ED    .2         MOVC3   #<DIRNAM-NAMBLK>,#0,NAMBLK(FP) ;REINIT NAME BLOCK
                A4 AD        00F1
                   3F    BA  00F3    .3         POPR    #^M<R0,R1,R2,R3,R4,R5>      ; RESTORE REGISTERS
-2                 56    D5  00F5   453          TSTL    R6                          ;ANY MORE DIRECTORY NAMES?
                   DF    14  00F7   454          BGTR    70$                         ;BRANCH IF YES, LOOKUP THE NEXT
          98 AD    52    7D  00F9   455          MOVQ    R2,NAMDSC(FP)               ;DESCRIPTOR FOR FILE TO LOOKUP
                   52    D4  00FD   456          CLRL    R2                          ;STOP THE LOOKUP LOOP
                98 AD    D5  00FF   457          TSTL    NAMDSC(FP)                  ;ALREADY DONE?
                   DB    14  0102   458          BGTR    80$                         ;BRANCH IF NO, DO THE LAST ONE
             07    6C    D1  0104   459 85$:     CMPL    ARGCNT(AP),#RTRVPTRBUF/4    ;RETRIEVAL POINTERS DESIRED?
                   03    19  0107   460          BLSS    90$                         ;BRANCH IF NOT
             6E    02    C0  0109   461          ADDL    #2,(SP)                     ;ADDITIONAL ARGUMENTS ARE PRESENT
          05F4'CF  6E    FA  010C   462 90$:     CALLG   (SP),W^FIL$RDCHKFILHDR      ;READ AND CHECK FILE HEADER
                   04    11  0111   463 100$:    RET
                         0112   464
                         0112   465          .DSABL  LSB
```

```
                                0112   467          .SBTTL  FIL$CACHE_INIT - INIT FILEREAD CACHE
                                0112   468 ;++
                                0112   469 ; FUNCTIONAL DESCRIPTION:
                                0112   470 ;
                                0112   471 ;       CACHE_INIT PERFORMS THE INITIALIZATION FOR THE FILEREAD CACHE
                                0112   472 ;
                                0112   473 ; CALLING SEQUENCE:
                                0112   474 ;
                                0112   475 ;       CALLG   ARGLIST,FIL$CACHE_INIT
                                0112   476 ;
                                0112   477 ; INPUT PARAMETERS:
                                0112   478 ;
                                0112   479 ;       CHANADR(AP)             ADDRESS TO RETURN LONG WORD CHANNEL
                                0112   480 ;       FILNAM(AP)              ADDRESS OF DEVICE NAME STRING DESCRIPTOR
                                0112   481 ;                               THE DEVICE NAME MUST CONTAIN THE ":"
                                0112   482 ;                               IF THE ADDRESS IS 0, THE STRING IS NULL,
                                0112   483 ;                               OR THE NAME DOES NOT CONTAIN A ":", THE
                                0112   484 ;                               DEFAULT DEVICE NAME IS USED
                                0112   485 ;       CACHE_SIZE(AP)          SIZE IN BYTES OF FILEREAD CACHE
                                0112   486 ;       CACHE_ADR(AP)           ADDRESS OF FILEREAD CACHE
                                0112   487 ;       DIR_CACHE_CNT(AP)       NUMBER OF DIRECTORY CACHE ENTRIES
                                0112   488 ;       LBN_CACHE_CNT(AP)       NUMBER OF LBN CACHE ENTRIES
                                0112   489 ;
                                0112   490 ; IMPLICIT INPUTS:
                                0112   491 ;
                                0112   492 ;       NONE
                                0112   493 ;
                                0112   494 ; OUTPUT PARAMETERS:
                                0112   495 ;
                                0112   496 ;       R0 = ALWAYS SUCCESSFUL STATUS CODE
                                0112   497 ;
                                0112   498 ; IMPLICIT OUTPUTS:
                                0112   499 ;
                                0112   500 ;       FIL$GQ_CACHE QUAD WORD FILLED IN WITH SIZE AND ADDRESS OF CACHE
                                0112   501 ;
                                0112   502 ; COMPLETION CODES:
                                0112   503 ;
                                0112   504 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                                0112   505 ;
                                0112   506 ; SIDE EFFECTS:
                                0112   507 ;
                                0112   508 ;       NONE
                                0112   509 ;
                                0112   510 ;
                                0112   511 ; EQUATED SYMBOLS, OFFSETS FROM AP
                                0112   512 ;
00000004                        0112   513          CHANADR          =       4
00000008                        0112   514          FILNAM           =       8
0000000C                        0112   515          CACHE_SIZE       =       12
00000010                        0112   516          CACHE_ADR        =       16
00000014                        0112   517          DIR_CACHE_CNT    =       20
00000018                        0112   518          LBN_CACHE_CNT    =       24
                                0112   519 ;
                                0112   520 ;--
                                0112   521 FIL$CACHE_INIT::
                          0C3C  0112   522          .WORD   ^M<R2,R3,R4,R5,R10,R11>
                                0114   523
```

```
                                0114    524             ASSUME  CACHE_SIZE+4 EQ CACHE_ADR
     5A    0C AC     7D       0114    525             MOVQ    CACHE_SIZE(AP),R10      ;R10=SIZE, R11=ADR
00000218 8F     C3   0118    526             SUBL3   #FIL$C_SIZE,R10,R0      ;BYTES LEFT FOR DIR AND LBN CACHES
     50    5A                0011E
           60     19   0120    527             BLSS    100$                    ;BRANCH IF NOT ENOUGH CACHE SPACE
     6B    01     B0   0122    528             MOVW    #FIL$C_CACHE_ID,FIL$W_CACHE_ID(R11) ;SET CACHE ID
                         0125    529                                             ;ALLOWS MOVING CACHES BETWEEN FILEREAD'S
00000218 8F     D0   0125    530             MOVL    #FIL$C_SIZE,FIL$L_DIROFF(R11) ;BEGINNING OF DIR CACHE
       04 AB                012B
00000218 8F     D0   012D    531             MOVL    #FIL$C_SIZE,FIL$L_DIRNXT(R11) ;NEXT AVAILABLE SLOT IN DIR CACHE
       08 AB                0133
    14 AC      24   C5   0135    532             MULL3   #FIL$C_DIR_SIZE,DIR_CACHE_CNT(AP),R1 ;BYTE COUNT FOR DIR CACHE
               51               0139
     50        51   C2   013A    533             SUBL    R1,R0                   ;BYTE COUNT LEFT FOR LBN CACHE
               43   19   013D    534             BLSS    100$                    ;BRANCH IF NOT ENOUGH SPACE
00000218 8F     C1   013F    535             ADDL3   #FIL$C_SIZE,R1,FIL$L_DIRMAX(R11) ;END OF DIR CACHE
    0C AB      51               0145
                         0148    536
                         0148    537             ASSUME  FIL$L_DIRMAX EQ FIL$L_LBNOFF
  10 AB   0C AB     D0   0148    538             MOVL    FIL$L_LBNOFF(R11),FIL$L_LBNNXT(R11) ;NEXT LBN ENTRY TO ALLOCATE
    18 AC    09     78   014D    539             ASHL    #9,LBN_CACHE_CNT(AP),R1 ;BYTE COUNT IN LBN CACHE
               51               0151
     50        51   D1   0152    540             CMPL    R1,R0                   ;ENOUGH ROOM FOR WHOLE LBN CACHE
               08   15   0155    541             BLEQ    20$                     ;BRANCH IF YES
000001FF 8F     CB   0157    542             BICL3   #^X1FF,R0,R1                ;USE WHAT IS LEFT TRUNCATED
     51        50               015D
    10 AB      51   C1   015F    543 20$:        ADDL3   R1,FIL$L_LBNNXT(R11),FIL$L_LBNMAX(R11) ;END OF LBN CACHE
          14 AB                0163
           003D     30   0165    544             BSBW    ASSIGN_DEV              ;ASSIGN THE DEVICE
          17 50     E9   0168    545             BLBC    R0,100$                 ;BRANCH IF ERROR
          18 AB     DF   016B    546             PUSHAL  FIL$A_IXFHDR(R11)       ;ADDRESS TO READ INDEX FILE HEADER
             7E     D4   016E    547             CLRL    -(SP)                   ;UNUSED PARAMETER
          04 BC     DD   0170    548             PUSHL   @CHANADR(AP)            ;CHANNEL JUST ASSIGNED
025D'CF     03     FB   0173    549             CALLS   #3,W^FIL$MOUNT          ;MOUNT THE VOLUME. RETURN INDEX FILE HDR
          07 50     E9   0178    550             BLBC    R0,100$                 ;BRANCH IF ERROR
             5A     7D   017B    551             MOVQ    R10,FIL$GQ_CACHE        ;SAVE DESCRIPTOR OF CACHE
 00000000'EF                017D
     50    01     D0   0182    552 100$:       MOVL    S^#SS$_NORMAL,R0
                   04   0185    553             RET
```

```
                                0186   555          .SBTTL  FIL$CACHE_TRUNC - TRUNCATE FILEREAD CACHE
                                0186   556 ;++
                                0186   557 ; FUNCTIONAL DESCRIPTION:
                                0186   558 ;
                                0186   559 ;       CACHE_TRUNC TRUNCATES THE FILEREAD CACHE AND MAKES IT IMPOSSIBLE
                                0186   560 ; TO ADD MORE DIRECTORY CACHE OR DIRECTORY LBN ENTRIES TO IT.  IN EFFECT
                                0186   561 ; THIS ROUTINE TURNS THE CACHE INTO A READ-ONLY DATA BASE.
                                0186   562 ;
                                0186   563 ; CALLING SEQUENCE:
                                0186   564 ;
                                0186   565 ;       CALLG   ARGLIST,FIL$CACHE_TRUNC
                                0186   566 ;
                                0186   567 ; INPUT PARAMETERS:
                                0186   568 ;
                                0186   569 ;       NONE
                                0186   570 ;
                                0186   571 ; IMPLICIT INPUTS:
                                0186   572 ;
                                0186   573 ;       FIL$GQ_CACHE            DESCRIPTOR FOR THE CACHE
                                0186   574 ;
                                0186   575 ; OUTPUT PARAMETERS:
                                0186   576 ;
                                0186   577 ;       R0 = ALWAYS SUCCESSFUL STATUS CODE
                                0186   578 ;
                                0186   579 ; IMPLICIT OUTPUTS:
                                0186   580 ;
                                0186   581 ;       FIL$GQ_CACHE FILLED IN WITH ALTERED SIZE OF CACHE
                                0186   582 ;
                                0186   583 ; COMPLETION CODES:
                                0186   584 ;
                                0186   585 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                                0186   586 ;
                                0186   587 ; SIDE EFFECTS:
                                0186   588 ;
                                0186   589 ;       NONE
                                0186   590 ;
                                0186   591 ; EQUATED SYMBOLS
                                0186   592 ;
                                0186   593 ;
                                0186   594 ;--
                                0186   595
                                0186   596 FIL$CACHE_TRUNC::
                          0000  0186   597          .WORD   0
  00000004'EF   D0              0188   598          MOVL    FIL$GQ_CACHE+4,R0       ;ADDRESS OF THE CACHE
                50              018E
0C A0   08 A0   D0              018F   599          MOVL    FIL$L_DIRNXT(R0),FIL$L_DIRMAX(R0) ;NO NEW DIRECTORY CACHE ENTRIES
14 A0   10 A0   D0              0194   600          MOVL    FIL$L_LBNNXT(R0),FIL$L_LBNMAX(R0) ;NO MORE LBN BUFFERS
        10 A0   D0              0199   601          MOVL    FIL$L_LBNNXT(R0),FIL$GQ_CACHE   ;SET NEW SIZE OF CACHE
  00000000'EF                   019C
     50   01    D0              01A1   602          MOVL    S^#SS$_NORMAL,R0
                04              01A4   603          RET
```

```
                                01A5   605          .SBTTL  ASSIGN_DEV - ASSIGN A DEVICE AND RETURN A CHANNEL
                                01A5   606 ;++
                                01A5   607 ; FUNCTIONAL DESCRIPTION:
                                01A5   608 ;
                                01A5   609 ;       ASSIGN THE GIVEN DEVICE AND RETURN A CHANNEL NUMBER
                                01A5   610 ;       FOR BOOTING THIS VALUE IS A LONG WORD
                                01A5   611 ;
                                01A5   612 ; INPUTS:
                                01A5   613 ;
                                01A5   614 ;       FILNAM(AP)      = ADDRESS OF STRING DESCRIPTOR
                                01A5   615 ;                         IF DEVICE NAME IS PRESENT IT MUST HAVE ":"
                                01A5   616 ;       CHANADR(AP)     = ADDRESS TO RETURN LONG WORD OF CHANNEL NUMBER
                                01A5   617 ;
                                01A5   618 ; OUTPUTS:
                                01A5   619 ;
                                01A5   620 ;       R0,R1 ALTERED
                                01A5   621 ;       R2 = SIZE OF DEVICE NAME STRING (DEFAULT IF NOT IN NAME STRING)
                                01A5   622 ;            MAYBE 0 IF DEFAULT DEVICE STRING WAS NOT PRESENT
                                01A5   623 ;            THIS IS THE CASE WHEN BOOTSTRAPPING.
                                01A5   624 ;       R3 = ADDRESS OF DEVICE NAME STRING
                                01A5   625 ;            0 IF DEFAULT DEVICE STRING WAS NOT PRESENT
                                01A5   626 ;
                                01A5   627 ;--
                                01A5   628
                                01A5   629 ASSIGN_DEV:
                          50    D4   01A5   630          CLRL    R0                      ;ASSUME NULL DEFAULT DEVICE STRING
                      FE59 CF   D5   01A7   631          TSTL    W^FIL_GT_DDDEV          ;ADDRESS OF DEFAULT DEVICE COUNTED STRING
                          0A    13   01AB   632          BEQL    10$                     ;BRANCH IF NO DEFAULT DEVICE STRING
                00000000'EF     DE   01AD   633          MOVAL   FIL$GT_DDDEV,R1         ;GET THE ADDRESS
                          51          01B3
                       50   81   9A   01B4   634          MOVZBL  (R1)+,R0                ;SIZE OF DEFAULT DEVICE STRING
                            03   BB   01B7   635 10$:     PUSHR   #^M<R0,R1>              ;PUSH DEFAULT DEVICE DESCRIPTOR
                            52   7C   01B9   636          CLRQ    R2                      ;ASSUME NULL STRING DESCRIPTOR
                     50     08   AC   D0   01BB   637          MOVL    FILNAM(AP),R0           ;ADDRESS OF STRING DESCRIPTOR
                            11   13   01BF   638          BEQL    20$                     ;BRANCH IF NO NAME GIVEN
                       52   60   7D   01C1   639          MOVQ    (R0),R2                 ;R2 = SIZE, R3 = ADR OF FILE NAME STRING
                63     52   3A   3A   01C4   640          LOCC    #^A/:/,R2,(R3)          ;ANY DEVICE SPECIFIED?
                            08   13   01C8   641          BEQL    20$                     ;BRANCH IF NONE
                  04    AE   53   D0   01CA   642          MOVL    R3,4(SP)                ;ADDRESS OF DEVICE NAME
                6E    51   53   C3   01CE   643          SUBL3   R3,R1,(SP)              ;SIZE OF DEVICE NAME STRING
                       50   5E   D0   01D2   644 20$:     MOVL    SP,R0                   ;ADDRESS OF DEV NAME DESCRIPTOR
                                   01D5   645 ;            $ASSIGN_S (R0),@CHANADR(AP)     ;ASSIGN THE CHANNEL
-1                          0C   BA   01D5   646          POPR    #^M<R2,R3>              ;GET DEVICE NAME SIZE AND ADDRESS
                                05   01D7   647          RSB
```

```
                     01D8   649          .SBTTL  STORE3DIGITS - STORE 3 ASCII DIGITS
                     01D8   650 ;++
                     01D8   651 ; FUNCTIONAL DESCRIPTION:
                     01D8   652 ;
                     01D8   653 ;        STORE 3 DIGITS OF DIRECTORY STRING
                     01D8   654 ;
                     01D8   655 ; CALLING SEQUENCE:
                     01D8   656 ;
                     01D8   657 ;        BSBB    STORE3DIGITS
                     01D8   658 ;
                     01D8   659 ; INPUT:
                     01D8   660 ;
                     01D8   661 ;        R0 = NO. OF DIGITS TO PUT IN STRING
                     01D8   662 ;        R1 = ADDRESS + 1 OF RIGHT MOST DIGIT
                     01D8   663 ;        R2 = ADDRESS AT WHICH TO STORE 3 DIGITS
                     01D8   664 ;
                     01D8   665 ; OUTPUTS:
                     01D8   666 ;
                     01D8   667 ;        NONE
                     01D8   668 ;
                     01D8   669 ;--
                     01D8   670
                     01D8   671 STORE3DIGITS:
          03   50   D1  01D8   672          CMPL    R0,#3                   ;3 DIGITS OR LESS SPECIFIED?
               03   15  01DB   673          BLEQ    5$                      ;YES. BRANCH.
            0279   31  01DD   674          BRW     BADFILNAM               ;NO. DIRECTORY STRING BAD. EXIT
                       01E0   675                                          ;WITH ERROR.
    82  3030  8F   BO  01E0   676 5$:       MOVW    #^A/00/,(R2)+           ;BACKGROUND WITH ASCII 0
        82    30   90  01E5   677          MOVB    #^A/0/,(R2)+            ;
              03   11  01E8   678          BRB     20$                     ;START LOOP AT BOTTOM
        72    71   90  01EA   679 10$:      MOVB    -(R1),-(R2)             ;STORE BYTES LAST TO FIRST
                       01ED   680                                          ;LEAVING LEADING ASCII 0'S
          FA  50   F4  01ED   681 20$:      SOBGEQ  R0,10$                  ;LOOP ZERO OR MORE TIMES
                   05  01F0   682          RSB
```

```
                                01F1   684          .SBTTL  FORMDIRSTRING - GET A DIRECTORY STRING
                                01F1   685 ;++
                                01F1   686 ; FUNCTIONAL DESCRIPTION:
                                01F1   687 ;
                                01F1   688 ;       PULL THE FIRST DIRECTORY NAME OFF THE FRONT OF THE INPUT
                                01F1   689 ;       DIRECTORY STRING AND FORM THE FULL FILE NAME OF THE DIRECTORY
                                01F1   690 ;       TO LOOK UP.
                                01F1   691 ;
                                01F1   692 ; CALLING SEQUENCE:
                                01F1   693 ;
                                01F1   694 ;       BSBW    FORMDIRSTRING
                                01F1   695 ;
                                01F1   696 ; INPUTS:
                                01F1   697 ;
                                01F1   698 ;       R6 = SIZE OF DIRECTORY STRING
                                01F1   699 ;       R7 = ADDRESS OF DIRECTORY STRING
                                01F1   700 ;               THE STRING CONTAINS NO BRACKETS,
                                01F1   701 ;               IT MAY BE OF THE FORM 'DIR1.DIR2.DIR3...DIRN'
                                01F1   702 ;               THE FIRST AND ONLY ITEM MAY BE IN THE FORM GROUP,MEMBER
                                01F1   703 ;       DIRNAM(FP) = ADDRESS OF AREA TO BUILD THE NAME
                                01F1   704 ;
                                01F1   705 ; OUTPUTS:
                                01F1   706 ;
                                01F1   707 ;       R0 = SIZE OF DIRECTORY STRING
                                01F1   708 ;       R1 = ADDRESS OF DIRECTORY STRING
                                01F1   709 ;       R2,R3 PRESERVED
                                01F1   710 ;       R6,R7 UPDATED TO POINT AT THE REST OF THE STRING
                                01F1   711 ;--
                                01F1   712
                                01F1   713 FORMDIRSTRING:
                67    56    2E    3A  01F1   714          LOCC    #^A/./,R6,(R7)          ;FIND NEXT DIRECTORY STRING
                56    50    01    C3  01F5   715          SUBL3   #1,R0,R6                ;SIZE OF REST, SKIP THE "."
                                      01F9   716                                          ;-1 IF EMPTY
                50    51    57    C3  01F9   717          SUBL3   R7,R1,R0                ;BYTE COUNT OF DIRECTORY NAME
                      51    57    D0  01FD   718          MOVL    R7,R1                   ;ADDRESS OF DIRECTORY NAME
                57    01  A140    9E  0200   719          MOVAB   1(R1)[R0],R7            ;ADDRESS OF NEXT BYTE BEYOND "."
                            07    BB  0205   720          PUSHR   #^M<R0,R1,R2>           ;SAVE STRING DESCRIPTORS AND R2
                      27    50    D1  0207   721          CMPL    R0,#39                  ;LENGTH OF DIRECTORY STRING OKAY?
-1                          03    15  020A   722          BLEQ    5$                      ;YES. BRANCH.
                          024A    31  020C   723          BRW     BADFILNAM               ;NO. EXIT WITH ERROR.
                61    50    2C    3A  020F   724 5$:      LOCC    #^A/,/,R0,(R1)          ;GOOD STRING: ANY ","?
                            39    13  0213   725          BEQL    20$                     ;BRANCH IF NOT, RETURN THE DESCRIPTOR AS IS
                            50    DD  0215   726          PUSHL   R0                      ;SAVE REMAINING BYTE COUNT
                  04  AE    50    C3  0217   727          SUBL3   R0,4(SP),R0             ;BYTE COUNT TO LEFT OF ","
                            50          021B
                  52    CC  AD    DE  021C   728          MOVAL   DIRNAM(FP),R2           ;ADDRESS TO STORE FIRST 3 CHARS
                            B6    10  0220   729          BSBB    STORE3DIGITS            ;STORE THEM
                50    8E    01    C3  0222   730          SUBL3   #1,(SP)+,R0             ;COUNT OF CHARS TO RIGHT OF ","
                51    8E    8E    C1  0226   731          ADDL3   (SP)+,(SP)+,R1          ;ADR OF BYTE TO RIGHT OF LAST CHAR
                  52    CF  AD    DE  022A   732          MOVAL   DIRNAM+3(FP),R2         ;ADR TO STORE LAST 3 CHARS OF DIR NAME
                            A8    10  022E   733          BSBB    STORE3DIGITS            ;STORE THEM
                            04    BA  0230   734          POPR    #^M<R2>                 ;RESTORE SAVED R2
                      50    06    D0  0232   735          MOVL    #6,R0                   ;6 BYTES STRING SIZE
                51    CC  AD40    9E  0235   736 10$:     MOVAB   DIRNAM(FP)[R0],R1       ;POINT TO END OF STRING
                 5249442E 8F    D0  023A   737          MOVL    #^A/.DIR/,(R1)+         ;PUT TYPE IN STRING
                            81          0240
                61    313B 8F    B0  0241   738          MOVW    #^A/;1/,(R1)            ;AND VERSION AS WELL
```

```
                51    CC AD    9E  0246    739          MOVAB   DIRNAM(FP),R1           ;ADDRESS OF STRING
                   50    06    C0  024A    740          ADDL    #6,R0                  ;SIZE INCLUDES ".DIR;1"
                         05    05  024D    741          RSB
                         38    BB  024E    742 20$:     PUSHR   #^M<R3,R4,R5>          ;SAVE THESE FROM MOVC3
                               0250    743 ;
                               0250    744 ; 12(SP) = SIZE OF STRING, 16(SP) = ADDRESS
                               0250    745 ;
              10 BE    0C AE    28  0250    746          MOVC3   12(SP),@16(SP),DIRNAM(FP) ;MOVE NAME TO SCRATCH AREA
                       CC AD        0255
                         38    BA  0257    747          POPR    #^M<R3,R4,R5>          ;RESTORE REGISTERS
                         07    BA  0259    748          POPR    #^M<R0,R1,R2>
                         D8    11  025B    749          BRB     10$
```

```
                                025D    751          .SBTTL  MOUNT - MOUNT THE VOLUME, INIT FOR FILE LOOKUP
                                025D    752 ;++
                                025D    753 ; FUNCTIONAL DESCRIPTION:
                                025D    754 ;
                                025D    755 ;       MOUNT PERFORMS THE NECESSARY INITIALIZATION FOR FILE LOOKUP.
                                025D    756 ; IT READS THE HOME BLOCK, AND THEN RETURNS THE INDEX FILE HEADER TO THE
                                025D    757 ; SPECIFIED BUFFER.  THE INDEX FILE HEADER IS ALTERED BY RECORDING THE
                                025D    758 ; VIRTUAL BLOCK OFFSET REQUIRED TO TRANSLATE 'FILE NUMBER' TO INDEX FILE VBN
                                025D    759 ;
                                025D    760 ; CALLING SEQUENCE:
                                025D    761 ;
                                025D    762 ;       CALLG   ARGLIST,FIL$MOUNT
                                025D    763 ;
                                025D    764 ; INPUT PARAMETERS:
                                025D    765 ;
                                025D    766 ;       CHAN(AP)                CHANNEL ON WHICH DEVICE IS ASSIGNED
                                025D    767 ;       UNUSED                  2ND PARAMETER NOT USED
                                025D    768 ;       IXFHDR(AP)              ADDRESS TO RETURN INDEX FILE HEADER
                                025D    769 ;
                                025D    770 ; IMPLICIT INPUTS:
                                025D    771 ;
                                025D    772 ;       NONE
                                025D    773 ;
                                025D    774 ; OUTPUT PARAMETERS:
                                025D    775 ;
                                025D    776 ;       R0 = SYSTEM STATUS CODE
                                025D    777 ;
                                025D    778 ; IMPLICIT OUTPUTS:
                                025D    779 ;
                                025D    780 ;       NONE
                                025D    781 ;
                                025D    782 ; COMPLETION CODES:
                                025D    783 ;
                                025D    784 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                                025D    785 ;       SS$_FILESTRUCT          FILE STRUCTURE LEVEL NOT SUPPORTED
                                025D    786 ;       SS$_BADCHKSUM           CHECKSUM ERROR ON HOME BLOCK OR INDEX FILE HEADER
                                025D    787 ;       SS$_BADFILEHDR          INDEX FILE HEADER IS BAD
                                025D    788 ;
                                025D    789 ; SIDE EFFECTS:
                                025D    790 ;
                                025D    791 ;       NONE
                                025D    792 ;
                                025D    793 ;
                                025D    794 ; EQUATED SYMBOLS, OFFSETS FROM AP
                                025D    795 ;
                    00000004    025D    796          CHAN            =       4
                    0000000C    025D    797          IXFHDR          =       12
                                025D    798 ;
                                025D    799 ;--
                                025D    800
                                025D    801 FIL$MOUNT::
                        001C    025D    802          .WORD   ^M<R2,R3,R4>
              53    0C AC   D0  025F    803          MOVL    IXFHDR(AP),R3           ;ADDRESS OF BUFFER
           7E    01    09   9C  0263    804          ROTL    #9,#1,-(SP)             ;NUMBER OF BYTES TO READ
                 7E    21   3C  0267    805          MOVZWL  #IO$_READLBLK,-(SP)     ;READ LOGICAL BLOCK FUNCTION
                       53   DD  026A    806          PUSHL   R3                      ;BUFFER ADDRESS
                       01   DD  026C    807          PUSHL   #1                      ;LOGICAL BLOCK NUMBER 1 IS HOME BLK
```

```
                04 AC    DD  026E    808          PUSHL   CHAN(AP)                        ;CHANNEL
                   05    DD  0271    809          PUSHL   #5                              ;NO. OF ARGUMENTS
          0000'CF   6E    FA  0273    810          CALLG   (SP),W^FIL$RDWRTLBN             ;READ THE HOME BLOCK
                71 50     E9  0278    811          BLBC    R0,30$                          ;BRANCH IF ERROR
             51    53     D0  027B    812          MOVL    R3,R1                           ;ADDRESS OF HOME BLOCK
             50    1D     3C  027E    813          MOVZWL  #HM1$W_CHECKSUM1a-1,R0          ;NO. OF WORDS IN FIRST CHECKSUM
                05B9     30  0281    814          BSBW    FIL$CHECKSUM1                   ;CHECK THE FIRST CHECKSUM
             51    53     D0  0284    815          MOVL    R3,R1                           ;ADR OF HOME BLOCK AGAIN
                05AE     30  0287    816          BSBW    FIL$CHECKSUM                    ;CHECK THE MAIN CHECKSUM
                         028A    817          CASE    HM1$W_STRUCLEV+1(R3),<-
                         028A    818                  15$,-                           ;STRUCTURE LEVEL 1
                         028A    819                  10$-                            ;STRUCTURE LEVEL 2
                         028A    820                  >,TYPE=B,LIMIT=#1
          50    08C0 8F    3C  0293    821 5$:      MOVZWL  #SS$_FILESTRUCT,R0              ;UNSUPPORTED FILE STRUCTURE LEVEL
                   04     0298    822          RET
                         0299    823 ;
                         0299    824 ; STRUCTURE LEVEL 2
                         0299    825 ;
             51    18 A3    D0  0299    826 10$:     MOVL    HM2$L_IBMAPLBN(R3),R1           ;INDEX BIT MAP STARTING LBN
             50    20 A3    3C  029D    827          MOVZWL  HM2$W_IBMAPSIZE(R3),R0          ;INDEX BIT MAP SIZE IN BLOCKS
             54    1C A3    D0  02A1    828          MOVL    HM2$L_MAXFILES(R3),R4           ;MAXIMUM FILES ON VOLUME
                         02A5    829                                                  ;ONLY INTERESETED IN HIGH 16 BITS
          0E A3    04    A5  02A5    830          MULW3   #4,HM2$W_CLUSTER(R3),R4         ;4*CLUSTER TO LOW WORD OF R4
                   54    02A9
                   0B    11  02AA    831          BRB     20$
                         02AC    832 ;
                         02AC    833 ; STRUCTURE LEVEL 1
                         02AC    834 ;
          02 A3    10    9C  02AC    835 15$:     ROTL    #16,HM1$L_IBMAPLBN(R3),R1       ;LOGICAL BLK NO. OF 1ST INDEX BIT MAP BLK
                   51        02B0
             50    63    3C  02B1    836          MOVZWL  HM1$W_IBMAPSIZE(R3),R0          ;NO. OF BLOCKS OF INDEX BIT MAP
             54    02    D0  02B4    837          MOVL    #2,R4                           ;NUMBER OF VBN'S BEFORE INDEX FILE BIT MAP
             54    50    A0  02B7    838 20$:     ADDW    R0,R4                           ;LOW WORD IS VBNOFFSET
                         02BA    839                                                  ;FROM FILE ID TO INDEX FILE VBN
                         02BA    840 ;
                         02BA    841 ; READ INDEX FILE HEADER
                         02BA    842 ;       R0 - NUMBER OF BLOCKS IN INDEX FILE
                         02BA    843 ;       R1 - STARTING LBN OF INDEX FILE
                         02BA    844 ;
             51    50    C1  02BA    845          ADDL3   R0,R1,8(SP)                     ;DESIRED LBN TO ARG LIST
                08 AE        02BD
          0000'CF   8E    FB  02BF    846          CALLS   (SP)+,W^FIL$RDWRTLBN            ;READ INDEX FILE HEADER
                         02C4    847                                                  ;STRIP OFF THE ARGUMENT LIST
                25 50    E9  02C4    848          BLBC    R0,30$                          ;BRANCH IF ERROR
             51    53     D0  02C7    849          MOVL    R3,R1                           ;ADDRESS OF HEADER
                   7E     D4  02CA    850          CLRL    -(SP)                           ;FORM FILE ID ON STACK
          00010001 8F     DD  02CC    851          PUSHL   #^X10001                        ;FOR THE INDEX FILE HEADER
             50    5E     D0  02D2    852          MOVL    SP,R0                           ;ADDRESS OF FILE ID
                0522     30  02D5    853          BSBW    FIL$CHKFILHDR                   ;CHECK THE FILE HEADER (SEE IF
                         02D8    854                                                  ;FILE IDS MATCH)
          01FE C3    54    B0  02D8    855          MOVW    R4,FH1$W_VBNOFFSET(R3)          ;STORE VBN OFFSET
                         02DD    856 ;
                         02DD    857 ; IF MAXFILES WAS GREATER THAN ^XFFFF THEN HIGH 16 BITS OF R4 WILL BE
                         02DD    858 ; NON-ZERO.  IN THIS CASE, RECORD THE BIGFILNUM BIT IN THE STRUCLEV WORD
                         02DD    859 ;
          54    F0 8F    78  02DD    860          ASHL    #-16,R4,R4                      ;SEE IF HIGH 16 BITS = 0
                   54        02E1
```

```
                05   13  02E2    861          BEQL    25$
 00 06 A3       0A   E2  02E4    862          BBSS    #FH2$V_BIGFILNUM,FH2$W_STRUCLEV(R3),25$ ;MUST USE HIGH 8 BITS
                                                                                              ;OF RVN FIELD AS FILE NUMBER EXTENSION
       50       01   3C  02E9    864 25$:     MOVZWL  #SS$_NORMAL,R0                          ;SUCCESSFUL COMPLETION
                     04  02EC    865 30$:     RET
```

```
                        867          .SBTTL  FINDFILID - FIND FILE ID FOR SPECIFIED FILE
                        868 ;++
                        869 ; FUNCTIONAL DESCRIPTION:
                        870 ;
                        871 ;       FINDFILID SCANS A SPECIFIED DIRECTORY FOR  FILE AND
                        872 ; RETURNS ITS FILE ID IF FOUND.  STRUCTURE LEVEL 1 AND 2 DIRECTORIES
                        873 ; ARE SUPPORTED, 0 VERSION NUMBER MEANS FIND MOST RECENT VERSION,
                        874 ; -1 VERSION (FIND OLDEST) IS NOT SUPPORTED.  VERSION FOUND IS RETURNED IN
                        875 ; THE NAME BLOCK ADDRESSED BY THE FILE DESCRIPTOR (LEVEL 1 ONLY).
                        876 ; NOTE THAT NON-CONTIGUOUS DIRECTORIES ARE NOT SUPPORTED.
                        877 ;
                        878 ; CALLING SEQUENCE:
                        879 ;
                        880 ;       CALLG   ARGLIST,FIL$FINDFILID
                        881 ;
                        882 ; INPUT PARAMETERS:
                        883 ;
                        884 ;       CHAN(AP)                ;CHANNEL ON WHICH DEVICE IS ASSIGNED
                        885 ;       FILDSC(AP)      =       ADDRESS OF 3 LONG WORD FILE DESCRIPTOR
                        886 ;                               1 - SIZE OF ASCII STRING,  A 0 VALUE MEANS
                        887 ;                                       USE THE CONTENTS OF THE NAMBLK BELOW
                        888 ;                               2 - ADDRESS OF ASCII STRING
                        889 ;                               3 - ADDRESS OF NAME BLOCK - USED ONLY FOR LEVEL 1
                        890 ;                                       MAY CONTAIN DEFAULTS, BUT MUST BE
                        891 ;                                       AT LEAST INITIALIZED TO ZERO
                        892 ;                                       IT WILL BE WRITTEN.
                        893 ;       IXFHDR(AP)              ADR OF INDEX FILE HDR AS RETURNED FROM FIL$MOUNT
                        894 ;       DIRBUF(AP)              ADR OF 512 BYTE BUFFER TO USE FOR DIRECTORY SCAN
                        895 ;       STATBLK(AP)             ADDRESS OF 2 LONG WORD AREA USED FOR A
                        896 ;                               SCRATCH STATISTICS BLOCK
                        897 ;       FILID(AP)               ADR OF 3 WORD AREA USED BOTH AS THE ID OF
                        898 ;                               THE DIRECTORY TO SCAN AND AS THE PLACE TO
                        899 ;                               RETURN THE ID OF THE FILE FOUND
                        900 ;
                        901 ; IMPLICIT INPUTS:
                        902 ;
                        903 ;       NONE
                        904 ;
                        905 ; OUTPUT PARAMETERS:
                        906 ;
                        907 ;       R0 = SYSTEM STATUS CODE
                        908 ;
                        909 ; IMPLICIT OUTPUTS:
                        910 ;
                        911 ;       NONE
                        912 ;
                        913 ; COMPLETION CODES:
                        914 ;
                        915 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                        916 ;       SS$_NOSUCHFILE          FILE NOT FOUND
                        917 ;       SS$_BADFILENAME         SYNTAX ERROR IN FILE NAME
                        918 ;       SS$_BADCHKSUM           CHECKSUM ERROR ON DIRECTORY FILE HEADER
                        919 ;       SS$_BADFILEHDR          DIRECTORY FILE HEADER WAS BAD
                        920 ;
                        921 ; SIDE EFFECTS:
                        922 ;
                        923 ;       NONE
```

```
                                02ED    924 ;
                                02ED    925 ;
                                02ED    926 ; EQUATED SYMBOLS, OFFSETS FROM AP
                                02ED    927 ;
                     00000004   02ED    928          CHAN            =       4
                     00000008   02ED    929          FILDSC          =       8
                     0000000C   02ED    930          IXFHDR          =       12
                     00000010   02ED    931          DIRBUF          =       16
                     00000014   02ED    932          STATBLK         =       20
                     00000018   02ED    933          FILID           =       24
                                02ED    934 ;
                                02ED    935 ; OFFSETS FROM FP
                                02ED    936 ;
                                02ED    937          $OFFSET 0,NEGATIVE,<-
                                C2ED    938          DIR_BFCNT,-                     ;BUFFER COUNT REMAINING IN LBN CACHE
                                02ED    939          DIR_BUF,-                       ;NEXT BUFFER ADDRESS IN DIR LBN CACHE
                                02ED    940          ENTRY_ADR,-                     ;FOUND CACHE ENTRY ADR
                                02ED    941          <ENTRY,FIL$C_DIR_SIZE>,-        ;CACHE ENTRY FOR SEARCH/CREATE
                                02ED    942          <SCRATCH_SIZE,0>=               ;SIZE OF SCRATCH AREA
                                02ED    943          >
                                FFFC            DIR_BFCNT:
                                FFF8            DIR_BUF:
                                FFF4            ENTRY_ADR:
                                FFD0            ENTRY:
                                FFD0            SCRATCH_SIZE:
                                02ED    944 ;
                                02ED    945 ;--
                                02ED    946
                                02ED    947 FIL$FINDFILID::
                           0FFC 02ED    948          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     5E    30   C2 02EF    949          SUBL    #-SCRATCH_SIZE,SP       ;ALLOCATE SCRATCH SPACE
                00   6E    00   2C 02F2    950          MOVC5   #0,(SP),#0,#-SCRATCH_SIZE,(SP) ;ZERO THE SCRATCH STORAGE
                     6E    30      02F6
                           5B   D4 02F8    951          CLRL    R11                     ;ASSUME NO CACHE
                     02    6C   D1 02FA    952          CMPL    (AP),#2                 ;IF ONLY 2 ARGUMENTS
                           08   12 02FD    953          BNEQ    5$
                  5B   04 AC   D0 02FF    954          MOVL    4(AP),R11               ;THE FIRST IS THE CACHE ADDRESS
                  5C   08 AC   D0 0303    955          MOVL    8(AP),AP                ;THE SECOND IS THE REAL ARGUMENT LIST
                           5B   D5 0307    956 5$:      TSTL    R11                     ;CACHE ENABLED?
                           66   13 0309    957          BEQL    65$                     ;BRANCH IF NOT
                                030B    958 ;
                                030B    959 ; WE DO HAVE A CACHE TO LOOK IN AND MAKE ENTRIES IN, SET UP THE
                                030B    960 ; SCRATCH REGION FOR A LOOKUP
                                030B    961 ;
                                030B    962          ASSUME  ENTRY EQ SCRATCH_SIZE   ;SCRATCH CACHE ENTRY MUST BE ON TOP
                                030B    963          ASSUME  FIL$A_DIR_FID EQ 0      ;DIR ID IS AT FRONT OF CACHE ENTRY
               18 BC    06   28 030B    964          MOVC3   #6,@FILID(AP),(SP)      ;STORE DIRECTORY ID
                     6E         030F
               50    08 BC   7D 0310    965          MOVQ    @FILDSC(AP),R0          ;GET LOOKUP NAME DESCRIPTOR
                     06   50   D1 0314    966          CMPL    R0,#6                   ;MUST BE MORE THAN ".DIR;1"
                          25   15 0317    967          BLEQ    10$                     ;BRANCH IF NOT A DIRECTORY NAME
                     0F   50   D1 0319    968          CMPL    R0,#15                  ;AT MOST 9 CHAR WITH ".DIR;1"
                          20   14 031C    969          BGTR    10$                     ;BRANCH IF NOT A DIRECTORY NAME
               52   51   50   C1 031E    970          ADDL3   R0,R1,R2                ;POINT OFF END OF NAME STRING
               72   313B 8F   B1 0322    971          CMPW    #^A/;1/,-(R2)           ;LAST 2 BYTES ";1" ?
                          15   12 0327    972          BNEQ    10$                     ;BRANCH IF NOT A DIRECTORY NAME
                5249442E  8F   D1 0329    973          CMPL    #^A/.DIR/,-(R2)         ;PRECEDED BY ".DIR" ?
```

```
                        72      032F
                        0C  12  0330   974          BNEQ    10$                             ;BRANCH IF NOT A DIRECTORY NAME
                     50 06  C2  0332   975          SUBL    #6,R0                           ;JUST KEEP THE NAME PART
                  D6 AD 50  90  0335   976          MOVB    R0,FIL$T_DIR_NAM+ENTRY(FP) ;PUT SIZE AND NAME STRING
                     61 50  28  0339   977          MOVC3   R0,(R1),FIL$T_DIR_NAM+1+ENTRY(FP) ;IN ENTRY TO LOOKUP
                        D7 AD       033C
                  5B    04 AB  C1  033E   978 10$:   ADDL3   FIL$L_DIROFF(R11),R11,R8 ;ADDRESS OF DIRECTORY CACHE
                           58       0342
                  5B    08 AB  C1  0343   979          ADDL3   FIL$L_DIRNXT(R11),R11,R9 ;ADDRESS OF LAST+1 BYTE
                           59       0347
                        1E  11  0348   980          BRB     60$                             ;LOOP 0 OR MORE TIMES
                                034A   981
                                034A   982          ASSUME  FIL$T_DIR_NAM EQ FIL$A_DIR_FID+6
                  D0 AD 10  29  034A   983 20$:     CMPC3   #6+10,FIL$A_DIR_FID+ENTRY(FP), - ;DOES FID AND NAME MATCH?
                        68      034E
                                034F   984                  FIL$A_DIR_FID(R8)
                        06  12  034F   985          BNEQ    30$                             ;BRANCH IF DIDN'T MATCH ALL OF IT
                  F4 AD 58  D0  0351   986          MOVL    R8,ENTRY_ADR(FP)                ;RECORD THAT A MATCH WAS FOUND
                        1E  11  0355   987          BRB     70$
                                0357   988 ;
                                0357   989 ; FAILED TO MATCH THE ENTIRE ENTRY, DID WE MATCH THE DIR ID FIELD?
                                0357   990 ;
                     0A 50  D1  0357   991 30$:     CMPL    R0,#10                          ;IF 10 OR LESS CHAR'S LEFT
                        09  14  035A   992          BGTR    50$                             ;THEN MATCHED THE DIR ID
                  F4 AD 58  D0  035C   993          MOVL    R8,ENTRY_ADR(FP)                ;SAVE THIS PARTIAL MATCH
                     D6 AD 95  0360   994          TSTB    FIL$T_DIR_NAM+ENTRY(FP) ;IF NOT SEARCHING FOR A DIR NAME
                        10  13  0363   995          BEQL    70$                             ;THEN THIS ENTRY WILL DO FINE
                     58 24  C0  0365   996 50$:     ADDL    #FIL$C_DIR_SIZE,R8              ;ADDRESS OF NEXT CACHE ENTRY
                     59 58  D1  0368   997 60$:     CMPL    R8,R9                           ;DONE SCANNING DIR CACHE?
                        DD  1F  036B   998          BLSSU   20$                             ;BRANCH IF NOT, CHECK NEXT ENTRY
                                036D   999 ;
                                036D  1000 ; ENTRY_ADR(FP) = ADDRESS OF CACHE HIT ENTRY
                                036D  1001 ;                 = 0 IF NO MATCH FOUND
                                036D  1002 ; IF WE DROP THROUGH TO HERE AND WE GOT A CACHE HIT, THEN IT WAS
                                036D  1003 ; NOT EXACTLY WHAT WE WERE LOOKING FOR.  BUT IT DID MATCH THE DIRECTORY.
                                036D  1004 ;
               58    F4 AD  D0  036D  1005          MOVL    ENTRY_ADR(FP),R8                ;WAS THERE A CACHE HIT?
                        45  13  0371  1006 65$:     BEQL    READ_DIR_HEADER                 ;BRANCH IF NO
                        0F  11  0373  1007          BRB     80$                             ;YES, FOR DIRECTORY LBN AND SIZE
                                0375  1008 ;
                                0375  1009 ; FOUND WHAT WE WERE LOOKING FOR - MAY ONLY NEED DIRECTORY LBN AND SIZE
                                0375  1010 ;
                     1E A8  D5  0375  1011 70$:     TSTL    FIL$A_DIR_OFID(R8)              ;DID WE GET A FILE ID?
                        0A  13  0378  1012          BEQL    80$                             ;BRANCH IF NOT
               1E A8    06  28  037A  1013          MOVC3   #6,FIL$A_DIR_OFID(R8),@FILID(AP) ;RETURN THE FILE ID
                     18 BC       037E
                  50    01  3C  0380  1014          MOVZWL  S^#SS$_NORMAL,R0                ;SET SUCCESS STATUS
                        04  0383  1015 72$:     RET                                     ;AND RETURN
                                0384  1016 ;
                                0384  1017 ; CACHE HIT ONLY FOUND THE DIRECTORY LBN AND SIZE, SAVING THE
                                0384  1018 ; READ OF THE DIRECTORY FILE HEADER.
                                0384  1019 ;
            E0 AD    10 A8  7D  0384  1020 80$:     MOVQ    FIL$Q_DIR_HDR(R8),FIL$Q_DIR_HDR+ENTRY(FP) ;SAVE DIRHDR
                                0389  1021                                             ;INFO FOR MAKING A NEW ENTRY
                                0389  1022                                             ;WITH THE DIRECTORY FID IN IT
            E8 AD    18 A8  D0  0389  1023          MOVL    FIL$L_DIR_BFOFF(R8),FIL$L_DIR_BFOFF+ENTRY(FP)
            EC AD    1C A8  B0  038E  1024          MOVW    FIL$W_DIR_BFCNT(R8),FIL$W_DIR_BFCNT+ENTRY(FP)
```

```
                                0393  1025                                    ;SAVE DIR LBN CACHE INFO TOO
                                0393  1026 ;
                                0393  1027 ; AT THIS POINT ENTRY(FP) CONTAINS DIRECTORY LBN CACHE INFORMATION
                                0393  1028 ; IF ONE HAD ALREADY EXISTED OR IF WE JUST CREATED IT.
                                0393  1029 ; SET UP THE WORKING LOCATIONS FOR THE DIRECTORY LBN CACHE
                                0393  1030 ;
   E4 AD    01    C3           0393  1031 90$:    SUBL3   #1,FIL$L_DIR_LBN+ENTRY(FP),R6 ;R6=STARTING LBN - 1
                56              0397
   57    E0 AD    3C           0398  1032         MOVZWL  FIL$W_DIR_BKCNT+ENTRY(FP),R7 ;R7=SIZE OF DIRECTORY IN BLOCKS
FC AD    EC AD    3C           039C  1033         MOVZWL  FIL$W_DIR_BFCNT+ENTRY(FP),DIR_BFCNT(FP) ;BUFFER COUNT IN LBN CACHE
   5B    E8 AD    C1           03A1  1034         ADDL3   FIL$L_DIR_BFOFF+ENTRY(FP),R11,DIR_BUF(FP) ;STARTING ADR IN CACHE
         F8 AD                 03A5
      57    56    C0           03A7  1035         ADDL    R6,R7                   ;LAST LBN OF FILE INCLUSIVE
   03 E2 AD       E9           03AA  1036         BLBC    FIL$B_DIR_LVL+ENTRY(FP),100$ ;BRANCH IF STRUCTURE LEVEL 1
         00BA     31           03AE  1037         BRW     FIND_LEVEL2_1
         0192     31           03B1  1038 100$:   BRW     FIND_LEVEL1_1
                               03B4  1039 BADDIR2:
         009C     31           03B4  1040         BRW     BADDIR
                               03B7  1041 BADRET1:
                  04           03B7  1042         RET
                               03B8  1043 ;
                               03B8  1044 ; CACHE WAS NOT ENABLED OR THERE WAS NOT A HIT FOR THIS DIRECTORY
                               03B8  1045 ;
                               03B8  1046 READ_DIR_HEADER:
            6C     FA           03B8  1047         CALLG   (AP),FIL$RDCHKFILHDR    ;READ AND CHECK DIRECTORY FILE HEADER
000005F4'EF                    03BA
         F5 50    E9           03BF  1048         BLBC    R0,BADRET1              ;BRANCH IF ERROR
55    10 AC       D0           03C2  1049         MOVL    DIRBUF(AP),R5           ;ADDRESS OF BUFFER TO READ DIRECTORY BLOCKS
14 BC    01       C3           03C6  1050         SUBL3   #1,@STATBLK(AP),R6      ;GET START LBN - 1
            56                 03CA
                               03CB  1051 ;
                               03CB  1052 ; IF THIS RESULT IS NEGATIVE, THEN THE DIRECTORY WAS NOT CONTIGUOUS.
                               03CB  1053 ; THIS CODE SUPPORTS ONLY CONTIGUOUS DIRECTORIES, ANOTHER BUFFER WOULD
                               03CB  1054 ; BE REQUIRED TO HOLD THE DIRECTORY HEADER IN ORDER TO SUPPORT NON-CONTIGUOS
                               03CB  1055 ; DIRECTORIES.  SUCH DIRECTORIES ARE ONLY CREATED BY FILES-11 WHEN
                               03CB  1056 ; A DIRECTORY MUST BE EXTENDED AND THERE IS NOT ENOUGH CONTIGUOUS SPACE
                               03CB  1057 ; ANYWHERE ON THE VOLUME TO MAKE A NEW DIRECTORY OF THE CORRECT SIZE.
                               03CB  1058 ;
            E7     19           03CB  1059         BLSS    BADDIR2                 ;BRANCH IF NOT CONTIGUOUS
                               03CD  1060 ;
                               03CD  1061 ; SEE IF THIS LOOKS LIKE A DIRECTORY FILE
                               03CD  1062 ;
54    0E A5      DE           03CD  1063         MOVAL   FH1$W_RECATTR(R5),R4    ;ADDRESS OF LEVEL 1 RECORD ATTRIBUTES
07 A5    01       83           03D1  1064         SUBB3   #1,FHT$W_STRUCLEV+1(R5),R10 ;0 IF LEVEL 1, 1 IF LEVEL 2
            5A                 03D5
            04     13           03D6  1065         BEQL    10$                     ;BRANCH IF LEVEL 1
54    14 A5      DE           03D8  1066         MOVAL   FH2$W_RECATTR(R5),R4    ;ADDRESS OF LEVEL 2 RECORD ATTRIBUTES
08 A4    10       9C           03DC  1067 10$:    ROTL    #16,FAT$L_EFBLK(R4),R7  ;VBN OF DIRECTORY EOF
            57                 03E0
      0C A4      B5           03E1  1068         TSTW    FAT$W_FFBYTE(R4)        ;IF ZERO, EFBLK IS LAST+1 VBN
            02     12           03E4  1069         BNEQ    20$
            57     D7           03E6  1070         DECL    R7                      ;CORRECT TO GET LAST VBN
   56       01     C1           03E8  1071 20$:    ADDL3   #1,R6,FIL$L_DIR_LBN+ENTRY(FP) ;SAVE START LBN,
      E4 AD                    03EB
E0 AD    57       B0           03ED  1072         MOVW    R7,FIL$W_DIR_BKCNT+ENTRY(FP) ;DIRECTORY SIZE,
E2 AD    5A       90           03F1  1073         MOVB    R10,FIL$B_DIR_LVL+ENTRY(FP) ;AND STRUCTURE LEVEL
                               03F5  1074 ;
```

```
                                03F5  1075 ; SEE IF WE CAN SET UP A CACHE OF THE DIRECTORY BLOCKS FOR THIS DIRECTORY
                                03F5  1076 ;
                          5B    D5  03F5  1077         TSTL    R11                             ;ANY CACHEING ENABLED?
                          51    13  03F7  1078         BEQL    80$                             ;BRANCH IF NOT
                14 AB     10 AB C3  03F9  1079         SUBL3   FIL$L_LBNNXT(R11),FIL$L_LBNMAX(R11),R2 ;NO. OF BYTES
                          52        03FE
                                    03FF  1080                                                 ;AVAILABLE TO ALLOCATE
                   52     F7 8F     78  03FF  1081         ASHL    #-9,R2,R2                   ;NO. OF PAGES AVAILABLE
                          52        0403
                          44    13  0404  1082         BEQL    80$                             ;BRANCH IF NO SPACE AT ALL
                      57  52    D1  0406  1083         CMPL    R2,R7                           ;ENOUGH ROOM FOR WHOLE DIR
                          03    15  0409  1084         BLEQ    40$                             ;BRANCH IF NOT, USE WHAT IS LEFT
                   52     57    D0  040B  1085         MOVL    R7,R2                           ;YES, USE THE RIGHT SIZE
                53    10 AB    D0  040E  1086 40$:     MOVL    FIL$L_LBNNXT(R11),R3            ;OFFSET TO DIR LBN CACHE
                                0412  1087 ;
                                0412  1088 ; READ THE DISK BLOCKS INTO THE LBN CACHE.
                                0412  1089 ;
             7E    52    09    78  0412  1090         ASHL    #9,R2,-(SP)                     ;BYTE COUNT TO TRANSFER
                   7E    21    3C  0416  1091         MOVZWL  #IO$_READLBLK,-(SP)             ;READ LOGICAL BLOCK FUNCTION
                      6B43    9F  0419  1092         PUSHAB  (R11)[R3]                       ;BUFFER ADDRESS
                     E4 AD    DD  041C  1093         PUSHL   FIL$L_DIR_LBN+ENTRY(FP)         ;STARTING LBN
                     04 AC    DD  041F  1094         PUSHL   CHAN(AP)                        ;CHANNEL
                        05    FB  0422  1095         CALLS   #5,FIL$RDWRTLBN                 ;FILL THE DIR LBN CACHE
                00000000'EF         0424
                     1E 50    E9  0429  1096         BLBC    R0,80$                          ;BRANCH IF ERROR
                                042C  1097 ;
                                042C  1098 ; NOTE THAT DIRECTORY BLOCKS ARE IN MEMORY
                                042C  1099 ;
                FC AD    52    D0  042C  1100         MOVL    R2,DIR_BFCNT(FP)                ;COUNT OF BLOCKS READ IN
             F8 AD     6B43    9E  0430  1101         MOVAB   (R11)[R3],DIR_BUF(FP)           ;ADDRESS OF FIRST BLOCK READ IN
                    D6 AD    95  0435  1102         TSTB    FIL$T_DIR_NAM+ENTRY(FP)         ;ARE WE LOOKING UP ANOTHER DIRECTORY?
                       10    12  0438  1103         BNEQ    80$                             ;BRANCH IF YES, DON'T ALLOCATE
                                043A  1104                                                 ;A PERMANENT CACHE ENTRY FOR
                                043A  1105                                                 ;AN INTERMEDIATE LEVEL DIRECTORY
                EC AD    52    B0  043A  1106         MOVW    R2,FIL$W_DIR_BFCNT+ENTRY(FP)    ;SET UP FOR PERMANENT ENTRY
                E8 AD    53    D0  043E  1107         MOVL    R3,FIL$L_DIR_BFOFF+ENTRY(FP)    ;SAVE THE SIZE AND OFFSET OF CACHE
             51    52    09    78  0442  1108         ASHL    #9,R2,R1                        ;NO. OF BYTES IN CACHE
                10 AB    51    C0  0446  1109         ADDL    R1,FIL$L_LBNNXT(R11)            ;ALLOCATE THE CACHE
                                044A  1110 ;
                                044A  1111 ; IF IT WAS POSSIBLE TO READ THE DIRECTORY INTO THE LBN CACHE AREA,
                                044A  1112 ; THE SCAN WILL FIND THESE BLOCKS AS THEY ARE NEEDED.
                                044A  1113 ;
                                044A  1114 80$:
                                044A  1115 ;
                                044A  1116 ; R6 = STARTING LBN - 1 FOR THE DIRECTORY
                                044A  1117 ; R7 = COUNT OF BLOCKS OF DIRECTORY TO BE SCANNED
                                044A  1118 ; R10[0:7] = 0 IF STRUCTURE LEVEL 1
                                044A  1119 ;            = 1 IF STRUCTURE LEVEL 2
                                044A  1120 ;
                   57    56    C0  044A  1121         ADDL    R6,R7                           ;LAST LBN OF FILE (INCLUSIVE)
                     0F 5A    E8  044D  1122         BLBS    R10,FIND_LEVEL2                 ;BRANCH IF STRUCTURE LEVEL 2
                      00E8    31  0450  1123         BRW     FIND_LEVEL1
                                0453  1124
                                0453  1125 ;
                                0453  1126 ; ERROR RETURNS.
                                0453  1127 ;
             50    0828 8F    3C  0453  1128 BADDIR: MOVZWL  #SS$_BADIRECTORY,R0
```

```
                        04  0458  1129 BADRET: RET
                            0459  1130 BADFILNAM:
              50  0818 8F   3C  0459  1131         MOVZWL  #SS$_BADFILENAME,R0     ;RETURN ERROR CODE.
                        04  045E  1132         RET
```

```
                                045F  1134          .SBTTL  FIL$FINDFILID - STRUCTURE LEVEL 2
                                045F  1135 ;
                                045F  1136 ; STRUCTURE LEVEL 2
                                045F  1137 ;
                                045F  1138 FIND_LEVEL2:
          EF 34 A5    0D    E1  045F  1139          BBC     #FH2$V_DIRECTORY,FH2$L_FILECHAR(R5),BADDIR ;DIRECTORY BIT MUST BE SE
                                0464  1140
                                0464  1141          ASSUME  FAT$B_RATTRIB EQ FAT$B_RTYPE+1
                0802 8F     B1  0464  1142          CMPW    #<FAT$M_NOSPAN @ 8 + FAT$C_VARIABLE>,- ;VARIABLE LENGTH
                       64       0468  1143                  FAT$B_RTYPE(R4)         ;RECORDS NOT CROSSING BLOCK BOUNDARIES
                          E8    0469  1144          BNEQ    BADDIR                  ;BRANCH IF BAD RECORD ATTRIBUTES
                                046B  1145 ;
                                046B  1146 ; ***** NOTE THAT EACH BLOCK MUST END IN A RECORD SIZE OF -1
                                046B  1147 ; ***** A RECORD IS NOT ALLOWED TO EXACTLY FILL THE BLOCK
                                046B  1148 ; ***** PDP-11 FILE CONTROL SERVICES WILL READ THIS FILE CORRECTLY, BUT
                                046B  1149 ; ***** WILL NOT WRITE IT PROPERLY.  LIKEWISE FOR RMS-11 AND RMS-32
                                046B  1150 ;
                                046B  1151 FIND_LEVEL2_1:
             58    08 BC    7D  046B  1152          MOVQ    @FILDSC(AP),R8          ;R8 = SIZE, R9 = ADDRESS OF FILE NAME STRING
                          5A    D4  046F  1153          CLRL    R10                     ;ASSUME DEFAULT VERSION
                   53     58    7D  0471  1154          MOVQ    R8,R3                   ;COPY FILE NAME DESCRIPTOR
             64    53     2E    3A  0474  1155          LOCC    #^A/./,R3,(R4)          ;FIND FILE TYPE DELIMITER IF PRESENT
                          07    13  0478  1156          BEQL    40$                     ;BRANCH IF NOT PRESENT
                   53     70    9E  047A  1157          MOVAB   -(R0),R3                ;SIZE OF REMAINING STRING
                54    01  A1    9E  047D  1158          MOVAB   1(R1),R4                ;ADDRESS OF STRING BEYOND DELIMITER
             64    53     3B    3A  0481  1159 40$:     LOCC    #^A/;/,R3,(R4)          ;SEE IF VERSION DELIMITER PRESENT
                          06    12  0485  1160          BNEQ    60$                     ;BRANCH IF IT IS
             64    53     2E    3A  0487  1161          LOCC    #^A/./,R3,(R4)          ;TRY ALTERNATE VERSION DELIMITER
                          34    13  048B  1162          BEQL    120$                    ;BRANCH IF NO VERSION STRING PRESENT
                                048D  1163 ;
                                048D  1164 ; R0 = BYTE COUNT OF VERSION STRING PLUS DELIMITER
                                048D  1165 ; R1 = ADDRESS OF VERSION DELIMITER
                                048D  1166 ;
                58    50    C2  048D  1167 60$:     SUBL    R0,R8                   ;REDUCE FILE NAME STRING SIZE
                                0490  1168                                          ;ELIMINATING VERSION STRING AND DELIMITER
                          7E    DF  0490  1169          PUSHAL  -(SP)                   ;RESERVE LONG WORD FOR VERSION NUMBER
                                0492  1170                                          ;AND PUSH ITS ADDRESS
                   01  A1    9F  0492  1171          PUSHAB  1(R1)                   ;ADDRESS OF VERSION STRING
                          70    9F  0495  1172          PUSHAB  -(R0)                   ;SIZE OF VERSION STRING
                          03    FB  0497  1173          CALLS   #3,LIB$CVT_DTB          ;CONVERT DECIMAL VERSION STRING TO BINARY
          00000000'EF             0499
                B8 50     E9  049E  1174          BLBC    R0,BADFILNAM            ;BRANCH IF SYNTAX ERROR IN VERSION STRING
             5A   8E     D0  04A1  1175          MOVL    (SP)+,R10               ;FETCH EXPLICIT VERSION NUMBER
                                04A4  1176 ;
                                04A4  1177 ; R6  = ADDRESS OF LAST LBN READ FROM DIRECTORY FILE (FIRST - 1)
                                04A4  1178 ; R7  = ADDRESS OF LAST LBN (INCLUSIVE) TO BE READ FROM DIRECTORY FILE
                                04A4  1179 ; R8  = SIZE OF NAME STRING TO SCAN FOR
                                04A4  1180 ; R9  = ADDRESS OF NAME STRING TO SCAN FOR
                                04A4  1181 ; R10 = FILE VERSION NUMBER IF EXPLICIT, OR 0 IF DEFAULT TO LATEST VERSION
                                04A4  1182 ;
                          1B    11  04A4  1183          BRB     120$                    ;BEGIN LOOP AT BOTTOM
                                04A6  1184 ;
                                04A6  1185 ; R5 = ADDRESS OF NEXT RECORD
                                04A6  1186 ;
             54    05 A5    9A  04A6  1187 100$:    MOVZBL  DIR$B_NAMECOUNT(R5),R4  ;GET SIZE OF "NAME.TYP" STRING
          00    69    58    2D  04AA  1188          CMPC5   R8,(R9),#0,R4,DIR$T_NAME(R5) ;SEE IF STRINGS MATCH
            06 A5    54        04AE
```

```
                1D   13   04B1  1189          BEQL    200$                            ;BRANCH IF THEY DO
                10   19   04B3  1190          BLSS    140$                            ;BRANCH IF BEYOND WHERE NAME WOULD GO
           50   65   3C   04B5  1191          MOVZWL  DIR$W_SIZE(R5),R0               ;USING THE SIZE OF THIS RECORD
      55   02 A540   9E   04B8  1192          MOVAB   2(R5)[R0],R5                    ;FORM ADDRESS OF NEXT RECORD
                65   B5   04BD  1193 110$:    TSTW    DIR$W_SIZE(R5)                  ;END OF BLOCK? (MARKED WITH -1)
                E5   14   04BF  1194          BGTR    100$                            ;BRANCH IF NOT
        06 56   57   F3   04C1  1195 120$:    AOBLEQ  R7,R6,160$
      50   0910 8F   3C   04C5  1196 140$:    MOVZWL  #SS$_NOSUCHFILE,R0              ;CANNOT FIND FILE
                04   04CA  1197 150$:    RET
                00F8 30   04CB  1198 160$:    BSBW    READ_DIR_LBN                    ;READ THE NEXT DIRECTORY LBN
                ED   11   04CE  1199          BRB     110$
                     04D0  1200 ;
                     04D0  1201 ; FOUND A MATCH OF FILE NAME AND TYPE
                     04D0  1202 ;
                54   D6   04D0  1203 200$:    INCL    R4                              ;ROUND UP NAME COUNT
           54   01   CA   04D2  1204          BICL    #1,R4                           ;TO EVEN NUMBER OF BYTES
      53   06 A544   9E   04D5  1205          MOVAB   DIR$T_NAME(R5)[R4],R3           ;ADDRESS OF FIRST VERSION ENTRY
           50   65   3C   04DA  1206          MOVZWL  DIR$W_SIZE(R5),R0               ;SIZE OF THIS RECORD
      55   02 A540   9E   04DD  1207          MOVAB   2(R5)[R0],R5                    ;FORM ADDRESS OF BEGINNING OF NEX RECORD
                     04E2  1208                                                      ;WHICH IS ALSO THE END OF THE VERSIONS
                5A   D5   04E2  1209          TSTL    R10                             ;LATEST VERSION DESIRED?
                11   13   04E4  1210          BEQL    240$                            ;BRANCH IF YES, R3 IS ADDRESS OF
                     04E6  1211                                                      ;DESIRED VERSION AND FILE ID
           63   5A   B1   04E6  1212 230$:    CMPW    R10,DIR$W_VERSION(R3)           ;IS THIS THE RIGHT VERSION?
                0C   13   04E9  1213          BEQL    240$                            ;BRANCH IF YES
                D8   1A   04EB  1214          BGTRU   140$                            ;BRANCH IF PAST WHERE IT WOULD BE
           53   08   C0   04ED  1215          ADDL    #DIR$C_VERSION,R3               ;NEXT VERSION ENTRY
           55   53   D1   04F0  1216          CMPL    R3,R5                           ;END OF RECORD?
                F1   1F   04F3  1217          BLSSU   230$                            ;BRANCH IF NOT, CHECK NEXT VERSION
                C6   11   04F5  1218          BRB     110$                            ;VERSION NOT IN THIS VERSION CHAIN
                     04F7  1219                                                      ;SEE IF IT IS IN THE NEXT RECORD
                     04F7  1220 ;
                     04F7  1221 ; FOUND THE FILE ID, RETURN IT TO CALLER
                     04F7  1222 ;
      56   02 A3   7D   04F7  1223 240$:    MOVQ    DIR$W_FID(R3),R6                ;GET THE FILE ID
        57   57   3C   04FB  1224          MOVZWL  R7,R7
      02 A3   06   28   04FE  1225          MOVC3   #6,DIR$W_FID(R3),@FILID(AP) ;AND RETURN IT TO THE CALLER
           18 BC         0502
                     0504  1226 ;
                     0504  1227 ; SEE IF WE SHOULD MAKE A CACHE ENTRY FOR THIS LOOKUP
                     0504  1228 ; R6,R7 = FID
                     0504  1229 ;
                     0504  1230 EXIT_FILID_FND:
                5B   D5   0504  1231          TSTL    R11                             ;IS THE CACHE ENABLED?
                2C   13   0506  1232          BEQL    100$                            ;BRANCH IF NOT, JUST RETURN THE FID
           D6 AD   95   0508  1233          TSTB    FIL$T_DIR_NAM+ENTRY(FP)         ;WAS LOOKUP FOR A DIRECTORY?
                07   12   050B  1234          BNEQ    20$                             ;BRANCH IF YES, MAKE A CACHE ENTRY
           F4 AD   D5   050D  1235          TSTL    ENTRY_ADR(FP)                   ;CACHE HIT FOR THIS DIR HDR?
                22   12   0510  1236          BNEQ    100$                            ;BRANCH IF YES
                08   11   0512  1237          BRB     30$                             ;MAKE THE CACHE ENTRY FOR THIS DIR HDR
   EE AD        56   D0   0514  1238 20$:     MOVL    R6,FIL$A_DIR_OFID+ENTRY(FP) ;STORE THE FID FOUND
   F2 AD        57   B0   0518  1239          MOVW    R7,FIL$A_DIR_OFID+4+ENTRY(FP)
   58     08 AB   D0   051C  1240 30$:     MOVL    FIL$L_DIRNXT(R11),R8            ;GET OFFSET TO FREE SPACE
50    58        24   C1   0520  1241          ADDL3   #FIL$C_DIR_SIZE,R8,R0           ;FORM OFFSET TO END OF NEW ENTRY
   0C AB        50   D1   0524  1242          CMPL    R0,FIL$L_DIRMAX(R11)            ;ENOUGH SPACE FOR NEW ENTRY?
                0A   14   0528  1243          BGTR    90$                             ;BRANCH IF NOT
   08 AB        50   D0   052A  1244          MOVL    R0,FIL$L_DIRNXT(R11)            ;YES, ALLOCATE THE NEW ENTRY
```

```
                DO AD   24   28  052E  1245          MOVC3   #FIL$C_DIR_SIZE,ENTRY(FP),(R11)[R8] ;AND WRITE IT
                    6B48         0532
                                 0534  1246 90$:
                   50   01   3C  0534  1247 100$:    MOVZWL  #SS$_NORMAL,R0          ;INDICATE SUCCESSFUL COMPLETION
                                04  0537  1248          RET
                                     0538  1249 ;
                                     0538  1250 ; BAD DIRECTORY FILE
                                     0538  1251 ;
                                     0538  1252 BADDIR1:
                        FF18   31    0538  1253          BRW     BADDIR
```

```
                                053B  1255          .SBTTL  FIL$FINDFILID - STRUCTURE LEVEL 1
                                053B  1256 ;
                                053B  1257 ; STRUCTURE LEVEL 1 FIND
                                053B  1258 ; R4 = ADDRESS OF RECORD ATTRIBUTES
                                053B  1259 ;
                                053B  1260 FIND_LEVEL1:
                   64   01   91 053B  1261          CMPB    #FAT$C_FIXED,FAT$B_RTYPE(R4) ;FIXED LENGTH RECORD?
                        F8   12 053E  1262          BNEQ    BADDIRT                      ;BRANCH IF NOT
                02 A4   10   B1 0540  1263          CMPW    #16,FAT$W_RSIZE(R4)          ;16 BYTES EACH
                        F2   12 0544  1264          BNEQ    BADDIR1                      ;BRANCH IF NOT
                                0546  1265 ;
                                0546  1266 ; GET FILE NAME BLOCK FOR STRUCTURE LEVEL 1 DIRECTORY SCAN
                                0546  1267 ;
                                0546  1268 FIND_LEVEL1_1:
                53   08 AC   D0 0546  1269          MOVL    FILDSC(AP),R3                ;ADDRESS OF 3 LONG WORD FILE DESCRIPTOR
                        63   D5 054A  1270          TSTL    (R3)                         ;SIZE OF ASCII STRING
                        0D   13 054C  1271          BEQL    25$                          ;BRANCH IF NAME BLOCK ALL SET UP
                     08 A3   DD 054E  1272          PUSHL   8(R3)                        ;ADDRESS OF NAME BLOCK
                        63   DF 0551  1273          PUSHAL  (R3)                         ;ADDRESS OF STRING DESCRIPTOR
           0000'CF      02   FB 0553  1274          CALLS   #2,W^FIL$CVTFILNAM           ;CONVERT ASCII STRING TO NAME BLOCK (RAD50)
                5D   50   E9 0558  1275          BLBC    R0,90$                       ;BRANCH IF ERROR
                53   08 A3   D0 055B  1276 25$:     MOVL    8(R3),R3                     ;ADDRESS OF NAME BLOCK
                6E   08 A3   3C 055F  1277          MOVZWL  8(R3),(SP)                   ;SAVE FILE VERSION OVER FILE SIZE
                                0563  1278                                               ;THAT WAS LEFT FROM THE STATBLK CALL
                50   18 AC   D0 0563  1279          MOVL    FILID(AP),R0                 ;ADDRESS OF RETURN FILE ID
                        80   D4 0567  1280          CLRL    (R0)+                        ;INIT TO ZERO
                        60   B4 0569  1281          CLRW    (R0)                         ;ALL 3 WORDS.
                        3D   11 056B  1282          BRB     80$                          ;START DIRECTORY SCAN AT BOTTOM OF LOOP
                                056D  1283 ;
                                056D  1284 ; THE FOLLOWING REGISTER CONVENTION ARE USED IN THE DIRECTORY SCAN
                                056D  1285 ;       R2 = CURRENT DIRECTORY ENTRY BEING SCANNED
                                056D  1286 ;       R3 = NAMEBLOCK TO COMPARE AGAINST
                                056D  1287 ;       R4 = ADDRESS OF LAST BYTE OF DIRECTORY BLOCK
                                056D  1288 ;       R5 = ADDRESS OF DIRECTORY BUFFER
                                056D  1289 ;       R6 = LOGICAL BLOCK NUMBER OF CURRENT DIRECTORY BLOCK
                                056D  1290 ;       R7 = LAST LOGICAL BLOCK NUMBER (INCLUSIVE) TO SCAN
                                056D  1291 ;       0(SP) = VERSION OF FILE FROM NAME BLOCK
                                056D  1292 ;
                     0056   30   056D  1293 30$:     BSBW    READ_DIR_LBN                 ;READ THE NEXT DIRECTORY LBN
             54   01FF C5   DE 0570  1294          MOVAL   511(R5),R4                   ;ADDRESS OF LAST BYTE OF BUFFER
                52   55   D0 0575  1295          MOVL    R5,R2                        ;SET DIRECTORY RECORD POINTER
                        62   B5 0578  1296 40$:     TSTW    (R2)                         ;IS THIS DIRECTORY RECORD EMPTY
                        28   13 057A  1297          BEQL    70$                          ;BRANCH IF YES
                  51    06 A2   DE 057C  1298          MOVAL   6(R2),R1                     ;STEP OVER FILE ID TO NAME PORTION
                50      53   D0 0580  1299          MOVL    R3,R0                        ;MAKE A SCRATCH COPY OF NAME BLOCK ADR
                81      80   D1 0583  1300          CMPL    (R0)+,(R1)+                  ;CHECK 1ST 2 WORDS OF RAD50 FILE NAME
                        1C   12 0586  1301          BNEQ    70$                          ;BRANCH IF NO MATCH
                81      80   D1 0588  1302          CMPL    (R0)+,(R1)+                  ;LAST WORD OF RAD50 FILE NAME
                                058B  1303                                               ;AND THE 1 WORD RAD50 FILE TYPE
                        17   12 058B  1304          BNEQ    70$                          ;BRANCH IF NO MATCH
                                058D  1305 ;
                                058D  1306 ; NOW CHECK THE VERSION, IF SPECIFIED VERSION WAS 0 WE'RE LOOKING FOR
                                058D  1307 ; THE LARGEST VERSION WHICH OTHERWISE MATCHES, KEEP THE CURRENT LARGEST
                                058D  1308 ; IN THE INPUT NAME BLOCK, THUS RETURNING THE VERSION NUMBER FOUND.
                                058D  1309 ;
                        6E   B5 058D  1310          TSTW    (SP)                         ;SEARCHING FOR HIGHEST VERSION?
                        07   13 058F  1311          BEQL    50$                          ;BRANCH IF YES
```

```
                61   60   B1  0591  1312          CMPW    (R0),(R1)               ;NO, JUST COMPARE FOR EXACT MATCH
                     0E   12  0594  1313          BNEQ    70$                     ;BRANCH IF NO MATCH
                     21   11  0596  1314          BRB     100$                    ;FILE FOUND, EXIT LOOP
                61   60   B1  0598  1315 50$:     CMPW    (R0),(R1)               ;HIGHER THAN PREVIOUS HIGHEST VERSION?
                     07   1E  059B  1316          BGEQU   70$                     ;BRANCH IF NOT
                60   61   B0  059D  1317          MOVW    (R1),(R0)               ;RECORD HIGHEST VERSION FOUND
             18 BC   62   D0  05A0  1318          MOVL    (R2),@FILID(AP)         ;AND RECORD ITS FILE ID
           52   10   54   F1  05A4  1319 70$:     ACBL    R4,#16,R2,40$           ;NEXT DIRECTORY RECORD IF ANY
                     FFCE   05A8
             BF 56   57   F3  05AA  1320 80$:     AOBLEQ  R7,R6,30$               ;NEXT DIRECTORY BLOCK IF ANY
                             05AE  1321 ;
                             05AE  1322 ; DIRECTORY COMPLETELY SCANNED, IF LOOKING FOR HIGHEST VERSION, THEN
                             05AE  1323 ; POSSIBLY A FILE WAS FOUND, OTHERWISE NO MATCH.
                             05AE  1324 ;
                   18 BC   D5  05AE  1325          TSTL    @FILID(AP)              ;IF ANY FILE ID WAS STORED
                             05B1  1326                                          ;THEN A FILE WAS FOUND
                      0A   12  05B1  1327          BNEQ    120$                    ;BRANCH IF FILE WAS FOUND
           50   0910 8F   3C  05B3  1328          MOVZWL  #SS$_NOSUCHFILE,R0      ;NO SUCH FILE
                        04   05B8  1329 90$:     RET
                             05B9  1330 ;
                             05B9  1331 ; EXACT MATCH, FILE FOUND
                             05B9  1332 ;
             18 BC   62   D0  05B9  1333 100$:    MOVL    (R2),@FILID(AP)         ;RETURN THE FILE ID
             56   18 BC   D0  05BD  1334 120$:    MOVL    @FILID(AP),R6           ;FETCH FILE ID
                     57   D4  05C1  1335          CLRL    R7                      ;INTO R6,R7
                   FF3E   31  05C3  1336          BRW     EXIT_FILID_FND
```

```
                                05C6  1338          .SBTTL  READ_DIR_LBN - READ NEXT DIRECTORY LBN
                                05C6  1339 ;++
                                05C6  1340 ; FUNCTIONAL DESCRIPTION:
                                05C6  1341 ;
                                05C6  1342 ;       READ THE NEXT DIRECTORY LBN FROM THE DISK OR POINT AT
                                05C6  1343 ;       THE CACHED COPY IF ONE IS PRESENT
                                05C6  1344 ;
                                05C6  1345 ; CALLING SEQUENCE:
                                05C6  1346 ;
                                05C6  1347 ;       BSBW    READ_DIR_LBN
                                05C6  1348 ;
                                05C6  1349 ; INPUT:
                                05C6  1350 ;
                                05C6  1351 ;       R6 = DESIRED LBN
                                05C6  1352 ;       DIRBUF(AP) = BUFFER ADDRESS TO READ IT INTO
                                05C6  1353 ;       CHAN(AP) = CHANNEL FOR FIL$RDWRTLBN
                                05C6  1354 ;       DIR_BFCNT(FP) = COUNT OF BUFFERS REMAINING IN DIR LBN CACHE
                                05C6  1355 ;       DIR_BUF(FP) = ADDRESS OF NEXT BUFFER IN DIR LBN CACHE
                                05C6  1356 ;
                                05C6  1357 ; OUTPUTS:
                                05C6  1358 ;
                                05C6  1359 ;       R5 = ADDRESS OF DESIRED DIRECTORY LBN
                                05C6  1360 ;       RSB IF SUCCESSFUL
                                05C6  1361 ;       RET WITH STATUS IN R0 IF ERROR
                                05C6  1362 ;       R0, R1 DESTROYED, OTHERS PRESERVED
                                05C6  1363 ;
                                05C6  1364 ;--
                                05C6  1365
                                05C6  1366 READ_DIR_LBN:
      FC AD    D5               05C6  1367          TSTL    DIR_BFCNT(FP)           ;ANYTHING LEFT IN DIR LB' CACHE?
         0E    13               05C9  1368          BEQL    20$                     ;BRANCH IF NOT
      FC AD    D7               05CB  1369          DECL    DIR_BFCNT(FP)           ;COUNT ANOTHER BUFFER USED
   55 F8 AD    D0               05CE  1370          MOVL    DIR_BUF(FP),R5          ;LOAD ADDRESS OF BUFFER
    0200 C5    DE               05D2  1371          MOVAL   512(R5),DIR_BUF(FP)     ;AND POINT TO NEXT BUFFER IF ANY
      F8 AD                     05D6
                05               05D8  1372 10$:     RSB
                                 05D9  1373 ;
                                 05D9  1374 ; DIR LBN CACHE RAN OUT OF BLOCKS OR NEVER HAD ANY AT ALL
                                 05D9  1375 ;
   55 10 AC     D0               05D9  1376 20$:     MOVL    DIRBUF(AP),R5           ;ADDRESS OF BUFFER TO READ INTO
                                 05DD  1377          READLBN CHAN(AP),R6,(R5)        ;READ THE DESIRED LBN
7E 01    09     9C               05DD                        ROTL    #9,#1,-(SP)
   7E    21     3C               05E1                        MOVZWL  #IO$_READLBLK,-(SP)
         65     DF               05E4                        PUSHAL  (R5)
         56     DD               05E6                        PUSHL   R6
      04 AC     DD               05E8                        PUSHL   CHAN(AP)
0000'CF 05      FB               05EB                        CALLS   #5,W^FIL$RDWRTLBN
      E5 50     E8               05F0  1378          BLBS    R0,10$                  ;BRANCH IF READ SUCCESSFULLY
                04               05F3  1379          RET                             ;RETURN ERROR STATUS
```

```
05F4  1381          .SBTTL  RDCHKFILHDR - READ AND CHECK FILE HEADER
05F4  1382 ;++
05F4  1383 ; FUNCTIONAL DESCRIPTION:
05F4  1384 ;
05F4  1385 ;       RDCHKFILHDR READS AND VALIDATES A FILE HEADER GIVEN ITS FILE ID
05F4  1386 ; AND THE INDEX FILE HEADER AS RETURNED BY FIL$MOUNT.
05F4  1387 ;
05F4  1388 ; CALLING SEQUENCE:
05F4  1389 ;
05F4  1390 ;       CALLG   ARGLIST,FIL$RDCHKFILHDR
05F4  1391 ;
05F4  1392 ; INPUT PARAMETERS:
05F4  1393 ;
05F4  1394 ;       CHAN(AP)                    CHANNEL ON WHICH DEVICE IS ASSIGNED
05F4  1395 ;       UNUSED                      UNUSED PARAMETER
05F4  1396 ;       IXFHDR(AP)                  ADR OF INDEX FILE HEADER AS RETURNED BY FIL$MOUNT
05F4  1397 ;       FILHDR(AP)                  ADDRESS OF 512 BYTE BUFFER FOR FILE HEADER
05F4  1398 ;       STATBLK(AP)                 ADR OF 2 LONG WORD AREA TO RETURN STATISTICS BLOCK
05F4  1399 ;       FILID(AP)                   ADDRESS OF 3 WORD FILE ID OF DESIRED FILE HEADER
05F4  1400 ;       RTRVPTRLEN(AP)  =           ADDRESS TO RETURN THE NUMBER OF
05F4  1401 ;                                   BYTES OF RETRIEVAL POINTERS STORED
05F4  1402 ;                                   ***** OPTIONAL PARAMETER *****
05F4  1403 ;       RTRVPTRBUF(AP)  =           ADDRESS OF RETRIEVAL POINTER
05F4  1404 ;                                   BUFFER DESCRIPTOR.  THIS PARAMETER
05F4  1405 ;                                   IS PRESENT IF AND ONLY IF
05F4  1406 ;                                   RTRVPTRLEN IS PRESENT.
05F4  1407 ;                                   THE RETRIEVAL POINTERS ARE RETURNED IN
05F4  1408 ;                                   THE FORM 32 BIT BLOCK COUNT, 32 BIT LBN
05F4  1409 ;                                   A ZERO BUFFER DESCRIPTOR ADDRESS OR A
05F4  1410 ;                                   ZERO BUFFER ADDRESS MEANS DON'T
05F4  1411 ;                                   RETURN RETRIEVAL POINTER INFO
05F4  1412 ;
05F4  1413 ; IMPLICIT INPUTS:
05F4  1414 ;
05F4  1415 ;       NONE
05F4  1416 ;
05F4  1417 ; OUTPUT PARAMETERS:
05F4  1418 ;
05F4  1419 ;       R0 = SYSTEM STATUS CODE
05F4  1420 ;
05F4  1421 ; IMPLICIT OUTPUTS:
05F4  1422 ;
05F4  1423 ;       NONE
05F4  1424 ;
05F4  1425 ; COMPLETION CODES:
05F4  1426 ;
05F4  1427 ;       SS$_NORMAL                  SUCCESSFUL COMPLETION
05F4  1428 ;
05F4  1429 ; SIDE EFFECTS:
05F4  1430 ;
05F4  1431 ;       NONE
05F4  1432 ;
05F4  1433 ;
05F4  1434 ; EQUATED SYMBOLS
05F4  1435 ;
05F4  1436 ;    OFFSETS FROM AP
05F4  1437 ;
```

```
                     00000000  05F4  1438          ARGCNT          =       0
                     00000004  05F4  1439          CHAN            =       4
                     0000000C  05F4  1440          IXFHDR          =       12
                     00000010  05F4  1441          FILHDR          =       16
                     00000014  05F4  1442          STATBLK         =       20
                     00000018  05F4  1443          FILID           =       24
                     0000001C  05F4  1444          RTRVPTRLEN      =       28      ; OPTIONAL PARAMETER
                     00000020  05F4  1445          RTRVPTRBUF      =       32      ; PRESENT IF AND ONLY IF RTRVPTRLEN IS
                               05F4  1446 ;
                               05F4  1447 ;    OFFSETS FROM FP
                               05F4  1448 ;
                               05F4  1449          $OFFSET 0,NEGATIVE,<-
                               05F4  1450          <HDRCNT>,-                      ;COUNT OF FILE HEADERS READ
                               05F4  1451          <TMPRTRVLEN>,-                  ;TEMP RTRV PTR BYTE COUNT
                               05F4  1452          <TMPRTRVDSC,8>-                 ;TEMP RTRV PTR BUFFER DESCRIPTOR
                               05F4  1453          >
                               FFFC          HDRCNT:
                               FFF8          TMPRTRVLEN:
                               FFF0          TMPRTRVDSC:
                               05F4  1454 ;
                               05F4  1455 ;--
                               05F4  1456
                               05F4  1457 FIL$RDCHKFILHDR::
                         007C  05F4  1458          .WORD   ^M<R2,R3,R4,R5,R6>
                   7E    01   CE  05F6  1459          MNEGL   #1,-(SP)                ;COUNT OF HEADER BLOCKS READ
                         7E   D4  05F9  1460          CLRL    -(SP)                   ;INIT RTRV PTR BYTE COUNT
                         7E   7C  05FB  1461          CLRQ    -(SP)                   ;ASSUME NO RTRV PTR BUFFER
                      08   6C   D1  05FD  1462          CMPL    ARGCNT(AP),#RTRVPTRBUF/4 ;RTRV PTR PARAMS PRESENT?
                           0D   19  0600  1463          BLSS    2$                      ;BRANCH IF NOT
                   50    20 AC   D0  0602  1464          MOVL    RTRVPTRBUF(AP),R0       ;RTRV BUFFER DESCRIPTOR ADDRESS
                           07   13  0606  1465          BEQL    2$                      ;BRANCH IF NONE SPECIFIED
                F0 AD      60   7D  0608  1466          MOVQ    (R0),TMPRTRVDSC(FP)     ;MAKE COPY OF BUF DESCRIPTOR
                     1C BC      D4  060C  1467          CLRL    @RTRVPTRLEN(AP)         ;INIT RETURNED BYTE COUNT
                                    060F  1468
                                    060F  1469          ASSUME  FILHDR EQ IXFHDR+4
                52     0C AC   7D  060F  1470 2$:       MOVQ    IXFHDR(AP),R2           ;R2 = INDEX FILE HEADER ADDRESS
                                    0613  1471                                          ;R3 = FILE HEADER ADDRESS
                                    0613  1472          ASSUME  FILID EQ STATBLK+4
                54     14 AC   7D  0613  1473          MOVQ    STATBLK(AP),R4          ;R4 = RETURN STATBLK ADDRESS
                                    0617  1474                                          ;R5 = ADDRESS OF FILE ID
                   7E    65   7D  0617  1475          MOVQ    (R5),-(SP)              ;COPY FILE ID TO WRITABLE SCRATCH
                   55    5E   D0  061A  1476          MOVL    SP,R5                   ;AND REMEMBER ITS ADDRESS
                   56    7E   7E  061D  1477          MOVAQ   -(SP),R6                ;RESERVE SCRATCH STATISTICS BLOCK
                                    0620  1478                                          ;AND SAVE ITS ADDRESS IN R6
                         64   7C  0620  1479          CLRQ    (R4)                    ;INIT THE RETURN STAT BLOCK
                         64   D7  0622  1480          DECL    (R4)                    ;-1 LBN MEANS NOT YET STORED
                   7E    65   3C  0624  1481 5$:       MOVZWL  (R5),-(SP)              ;FILE NUMBER TO LONG WORD
             05 06 A2    0A   E1  0627  1482          BBC     #FH2$V_BIGFILNUM,FH2$W_STRUCLEV(R2),10$ ;BRANCH IF NO BIG FIL NUM
             02 AE    05 A5   90  062C  1483          MOVB    FID$B_RVNX(R5),2(SP)    ;HIGH 8 BITS OF RVN COMPLETE FILE NUMBER
                   50    8E   D0  0631  1484 10$:      MOVL    (SP)+,R0                ;IF FILE ID IS ZERO,
                         03   12  0634  1485          BNEQ    12$
                            0081 31  0636  1486          BRW     40$                     ;THEN READ LAST HEADER BLOCK
                51    01FE C2   3C  0639  1487 12$:      MOVZWL  FH1$W_VBNOFFSET(R2),R1  ;RECOVER VBN OFFSET FROM INDEX FILE HEADER
                   50    51   C0  063E  1488          ADDL    R1,R0                   ;ADD VBN OFFSET TO FORM INDEX FILE VBN
                      FC AD   D6  0641  1489          INCL    HDRCNT(FP)              ;COUNT EACH HEADER READ
                                    0644  1490          READVBN CHAN(AP),R0,(R3),(R2)   ;READ THE FILE HEADER
                         62   DF  0644                  PUSHAL  (R2)
```

```
                63   DF   0646                          PUSHAL  (R3)
                50   DD   0648                          PUSHL   R0
              04 AC   DD   064A                          PUSHL   CHAN(AP)
 06D3'CF      04   FB   064D                          CALLS   #4,W^FIL$READVBN
            76 50   E9   0652  1491          BLBC    R0,50$                  ;BRANCH IF ERROR
 71 FC AD     1F   E0   0655  1492          BBS     #31,HDRCNT(FP),50$      ;BRANCH IF JUST RE-READING MAIN HEADER
          50   55   D0   065A  1493          MOVL    R5,R0                   ;GET FILE ID ADDRESS
          51   53   D0   065D  1494          MOVL    R3,R1                   ;ADDRESS OF FILE HEADER
             0197   30   0660  1495          BSBW    FIL$CHKFILHDR           ;CHECK THE FILE HEADER
    52    0C AC   D0   0663  1496          MOVL    IXFHDR(AP),R2           ;INDEX FILE HEADER ADDRESS
          F0 AD   DF   0667  1497          PUSHAL  TMPRTRVDSC(FP)          ;RTRV PTR BUF DESCRIPTOR
          F8 AD   DF   066A  1498          PUSHAL  TMPRTRVLEN(FP)          ;ADDRESS TO RETURN BYTE COUNT
                56   DD   066D  1499          PUSHL   R6                      ;ADDRESS OF SCRATCH STAT BLOCK
                53   DD   066F  1500          PUSHL   R3                      ;ADDRESS OF FILE HEADER
 078A'CF      04   FB   0671  1501          CALLS   #4,W^FIL$STATBLK        ;READ STATISTICS BLOCK
    51    F8 AD   D0   0676  1502          MOVL    TMPRTRVLEN(FP),R1       ;ANY RTRV PTR INFO TO RETURN?
                16   13   067A  1503          BEQL    16$                     ;ZERO IF NONE REQUESTED
    1C BC    51   C0   067C  1504          ADDL    R1,@RTRVPTRLEN(AP)      ;ACCUMULATE RTRVPTR BYTE COUNT
    F0 AD    51   D1   0680  1505          CMPL    R1,TMPRTRVDSC(FP)       ;MORE SPACE NEEDED THAN WOULD FIT?
                04   15   0684  1506          BLEQ    14$                     ;BRANCH IF NOT
    51    F0 AD   D0   0686  1507          MOVL    TMPRTRVDSC(FP),R1       ;SAY WE USED IT ALL UP
    F4 AD    51   C0   068A  1508 14$:      ADDL    R1,TMPRTRVDSC+4(FP)     ;GET NEW STARTING ADDRESS
    F0 AD    51   C2   068E  1509          SUBL    R1,TMPRTRVDSC(FP)       ;AND CALC NEW SIZE REMAINING
          51   64   D2   0692  1510 16$:      MCOML   (R4),R1                 ;SEE IF START LBN HAS BEEN SET
                03   12   0695  1511          BNEQ    20$                     ;BRANCH IF IT HAS
          64   66   D0   0697  1512          MOVL    (R6),(R4)               ;SET IT ONCE ONLY
 04 A4     04 A6   C0   069A  1513 20$:      ADDL    4(R6),4(R4)             ;ADD IN THE SIZE FROM THIS HEADER
    65    0E A3   7D   069F  1514          MOVQ    FH2$W_EXT_FID(R3),(R5)  ;GET EXTENSION FILE ID IF ANY
 0F 06 A3     09   E0   06A3  1515          BBS     #FH2$V_LEVEL2,FH2$W_STRUCLEV(R3),30$ ;GOT IT IF LEVEL2
    50    01 A3   9A   06A8  1516          MOVZBL  FH1$B_MPOFFSET(R3),R0   ;GET WORD OFFSET TO LEVEL1 MAP AREA
     50    6340   3E   06AC  1517          MOVAW   (R3)[R0],R0             ;ADDRESS OF MAP AREA
    65    02 A0   D0   06B0  1518          MOVL    FM1$W_EX_FILNUM(R0),(R5) ;GET EXTENSION FILE NUMBER IF ANY
          04 A5   D4   06B4  1519          CLRL    4(R5)                   ;ZERO RVN
           FF6A   31   06B7  1520 30$:      BRW     5$                      ;READ THIS HEADER IF ANY
                       06BA  1521 ;
                       06BA  1522 ; LAST FILE HEADER READ, SEE IF MUST RE-READ THE ORIGINAL HEADER
                       06BA  1523 ;
          FC AD   D5   06BA  1524 40$:      TSTL    HDRCNT(FP)              ;WAS =1, BUMPED ONCE PER READVBN
             09   15   06BD  1525          BLEQ    45$                     ;BRANCH IF STILL HAVE MASTER FILE HEADER
    65    18 BC   7D   06BF  1526          MOVQ    @FILID(AP),(R5)         ;ORIGINAL FILE ID AGAIN
 EF FC AD     1F   E3   06C3  1527          BBCS    #31,HDRCNT(FP),30$      ;SET SIGN BIT AND GO READ ORIGINAL HEADER
          50   01   3C   06C8  1528 45$:      MOVZWL  #SS$_NORMAL,R0          ;RETURN SUCCESS STATUS
                04   06CB  1529 50$:      RET
```

```
                    06CC  1531          .SBTTL  READVBN, WRITEVBN - READ/WRITE VIRTUAL BLOCK
                    06CC  1532 ;++
                    06CC  1533 ; FUNCTIONAL DESCRIPTION:
                    06CC  1534 ;
                    06CC  1535 ;       THESE ROUTINES READ OR WRITE A VIRTUAL BLOCK FROM A FILE.
                    06CC  1536 ; VOLUME IS SPECIFIED BY THE CHANNEL TO WHICH IT IS ASSIGNED, AND THE
                    06CC  1537 ; FILE IS SPECIFIED BY THE ADDRESS OF ITS FILE HEADER WHICH WAS PREVIOUSLY
                    06CC  1538 ; READ BY A CALL TO FIL$RDFILHDR.
                    06CC  1539 ;
                    06CC  1540 ; CALLING SEQUENCE:
                    06CC  1541 ;
                    06CC  1542 ;       CALLG   ARGLIST,FIL$READVBN
                    06CC  1543 ;       CALLG   ARGLIST,FIL$WRITEVBN
                    06CC  1544 ;
                    06CC  1545 ; INPUT PARAMETERS:
                    06CC  1546 ;
                    06CC  1547 ;       CHAN(AP)        =               ;CHANNEL TO WHICH VOLUME IS ASSIGNED
                    06CC  1548 ;       VBN(AP)         =               ;DESIRED VIRTUAL BLOCK NUMBER
                    06CC  1549 ;       BUFADR(AP)      =               ;ADDRESS OF BUFFER TO READ INTO
                    06CC  1550 ;       FILHDR(AP)      =               ;ADDRESS OF FILE HEADER
                    06CC  1551 ;
                    06CC  1552 ; IMPLICIT INPUTS:
                    06CC  1553 ;
                    06CC  1554 ;       NONE
                    06CC  1555 ;
                    06CC  1556 ; OUTPUT PARAMETERS:
                    06CC  1557 ;
                    06CC  1558 ;       R0 = SYSTEM STATUS CODE
                    06CC  1559 ;
                    06CC  1560 ; IMPLICIT OUTPUTS:
                    06CC  1561 ;
                    06CC  1562 ;       NONE
                    06CC  1563 ;
                    06CC  1564 ; COMPLETION CODES:
                    06CC  1565 ;
                    06CC  1566 ;       SS$_NORMAL                  SUCCESSFUL RETURN
                    06CC  1567 ;       SS$_ENDOFFILE               SPECIFIED VBN BEYOND END OF FILE
                    06CC  1568 ;
                    06CC  1569 ; SIDE EFFECTS:
                    06CC  1570 ;
                    06CC  1571 ;       NONE
                    06CC  1572 ;
                    06CC  1573 ;
                    06CC  1574 ; EQUATED SYMBOLS:
                    06CC  1575 ;
                    06CC  1576 ;    OFFSET FROM AP
                    06CC  1577 ;
00000004            06CC  1578          CHAN            =       4       ;CHANNEL TO WHICH VOLUME IS ASSIGNED
00000008            06CC  1579          VBN             =       8       ;VIRTUAL BLOCK NUMBER
0000000C            06CC  1580          BUFADR          =       12      ;BUFFER ADDRESS TO READ INTO
00000010            06CC  1581          FILHDR          =       16      ;ADDRESS OF FILE HEADER
                    06CC  1582 ;
                    06CC  1583 ;    OFFSETS FROM FP
                    06CC  1584 ;
FFFFFFFC            06CC  1585          IOFUNCTION      =       -4      ;SAVED I/O FUNCTION CODE
                    06CC  1586 ;
                    06CC  1587 ;--
```

```
                                06CC  1588
                                06CC  1589 FIL$WRITEVBN::
                          003C  06CC  1590          .WORD   ^M<R2,R3,R4,R5>
                     7E   20  3C  06CE  1591          MOVZWL  #IO$_WRITELBLK,-(SP)
                          05  11  06D1  1592          BRB     RDWRTVBN
                                06D3  1593
                                06D3  1594 FIL$READVBN::
                          003C  06D3  1595          .WORD   ^M<R2,R3,R4,R5>
                     7E   21  3C  06D5  1596          MOVZWL  #IO$_READLBLK,-(SP)
                                06D8  1597
                                06D8  1598 RDWRTVBN:
                55   10 AC   D0  06D8  1599          MOVL    FILHDR(AP),R5           ;BASE ADR OF FILE HEADER
                52   06 A5   3C  06DC  1600          MOVZWL  FH1$W_STRUCLEV(R5),R2   ;STRUCTURE LEVEL
                        31   10  06E0  1601          BSBB    INIRTRVPTRSCAN          ;SET UP TO SCAN RETRIEVAL POINTERS
                                06E2  1602 ;
                                06E2  1603 ; R4 = POINTER TO FIRST RETRIEVAL POINTER,
                                06E2  1604 ; R5 = POINTER TO FIRST BYTE BEYOND LAST RETRIEVAL POINTER
                                06E2  1605 ; LOOP THROUGH RETRIEVAL POINTERS TO FIND THE ONE WHICH CONTAINS THE DESIRED VBN
                                06E2  1606 ;
             08 AC      01   C3  06E2  1607          SUBL3   #1,VBN(AP),R3           ;VBN BASE 0 TO LOOK FOR
                        53       06E6
                        48   10  06E7  1608 20$:     BSBB    GETRTRVPTR              ;FETCH NEXT RETRIEVAL POINTER
                     50 53   D1  06E9  1609          CMPL    R3,R0                   ;IS VBN IN THIS RETRIEVAL POINTER
                        0E   19  06EC  1610          BLSS    40$                     ;BRANCH IF YES
                     53 50   C2  06EE  1611          SUBL    R0,R3                   ;PASS OVER THAT MANY VBN'S
                     55 54   D1  06F1  1612          CMPL    R4,R5                   ;ANY MORE RETRIEVAL POINTERS?
                        F1   1F  06F4  1613          BLSSU   20$                     ;BRANCH IF YES
          50    0870 8F   3C  06F6  1614          MOVZWL  #SS$_ENDOFFILE,R0       ;RETURN END OF FILE INDICATION
                        04   06FB  1615          RET
                                06FC  1616 ;
                                06FC  1617 ; VBN IS IN THIS RETRIEVAL POINTER, R1 = STARTING LBN
                                06FC  1618 ;
          7E    01      09   9C  06FC  1619 40$:     ROTL    #9,#1,-(SP)             ;NUMBER OF BYTES TO READ/WRITE
                   FC AD   DD  0700  1620          PUSHL   IOFUNCTION(FP)          ;FUNCTION CODE
                   0C AC   DD  0703  1621          PUSHL   BUFADR(AP)              ;BUFFER TO TRANSFER TO/FROM
          7E    51      53   C1  0706  1622          ADDL3   R3,R1,-(SP)             ;LBN
                   04 AC   DD  070A  1623          PUSHL   CHAN(AP)                ;CHANNEL
     0000'CF            05   FB  070D  1624          CALLS   #5,W^FIL$RDWRTLBN       ;TRANSFER THE BLOCK
                        04   0712  1625          RET
```

```
                        0713  1627          .SBTTL  INIRTRVPTRSCAN - INITIALIZE RETRIEVAL POINTER SCAN
                        0713  1628 ;++
                        0713  1629 ; FUNCTIONAL DESCRIPTION:
                        0713  1630 ;
                        0713  1631 ;       LOCATE START AND END OF RETRIEVAL POINTERS IN A FILE HEADER.
                        0713  1632 ;
                        0713  1633 ; CALLING SEQUENCE:
                        0713  1634 ;
                        0713  1635 ;       BSBW    INIRTRVPTRSCAN
                        0713  1636 ;
                        0713  1637 ; INPUT:
                        0713  1638 ;
                        0713  1639 ;       R2 = STRUCTURE LEVEL
                        0713  1640 ;       R5 = FILE HEADER ADDRESS
                        0713  1641 ;
                        0713  1642 ; OUTPUT:
                        0713  1643 ;
                        0713  1644 ;       R4 = ADDRESS OF 1ST RETRIEVAL POINTER
                        0713  1645 ;       R5 = ADDRESS OF FIRST BYTE BEYOND LAST RETREIVAL POINTER
                        0713  1646 ;
                        0713  1647 ;--
                        0713  1648
                        0713  1649 INIRTRVPTRSCAN:
 50   01 A5    9A      0713  1650          MOVZBL  FH1$B_MPOFFSET(R5),R0   ;WORD OFFSET TO MAP AREA
  54    6540    3E      0717  1651          MOVAW   (R5)[R0],R4             ;BASE ADR OF MAP AREA
0C 52    09    E0      071B  1652          BBS     #FH2$V_LEVEL2,R2,20$    ;BRANCH IF LEVEL2
55    08 A4    9A      071F  1653          MOVZBL  FM1$B_INUSE(R4),R5      ;WORDS OF RETRIEVAL POINTERS IN USE
   54    0A    C0      0723  1654          ADDL    #FM1$C_POINTERS,R4      ;BEGINNING OF RETRIEVAL POINTERS
 55    6445    3E      0726  1655 10$:      MOVAW   (R4)[R5],R5             ;ADR JUST BEYOND LAST VALID RTRV PTR
               05      072A  1656          RSB
55    3A A5    9A      072B  1657 20$:      MOVZBL  FH2$B_MAP_INUSE(R5),R5  ;NO. OF WORDS OF RTRV PTRS IN USE
         F5    11      072F  1658          BRB     10$
```

```
                                0731  1660          .SBTTL  GETRTRVPTR - CONVERT NEXT RETRIEVAL POINTER
                                0731  1661 ;++
                                0731  1662 ; FUNCTIONAL DESCRIPTION:
                                0731  1663 ;
                                0731  1664 ;       CONVERT NEXT RETRIEVAL POINTER TO NUMBER OF BLOCKS COVERED BY
                                0731  1665 ;       POINTER AND STARTING LBN.
                                0731  1666 ;
                                0731  1667 ; CALLING SEQUENCE:
                                0731  1668 ;
                                0731  1669 ;       BSBW    GETRTRVPTR
                                0731  1670 ;
                                0731  1671 ; INPUTS:
                                0731  1672 ;
                                0731  1673 ;       R2 = STRUCTURE LEVEL WORD
                                0731  1674 ;       R4 = ADDRESS OF NEXT RETRIEVAL POINTER
                                0731  1675 ;
                                0731  1676 ; OUTPUTS:
                                0731  1677 ;
                                0731  1678 ;       R0 = NUMBER OF BLOCKS COVERED BY THE RETRIEVAL POINTER
                                0731  1679 ;       R1 = STARTING LOGICAL BLOCK NUMBER
                                0731  1680 ;       R2,R3 PRESERVED
                                0731  1681 ;
                                0731  1682 ;--
                                0731  1683
                                0731  1684 GETRTRVPTR:
                    OF 52   09  E0  0731  1685          BBS     #FH2$V_LEVEL2,R2,20$    ;BRANCH IF STRUCTURE LEVEL 2
                                0735  1686 ;
                                0735  1687 ; STRUCTURE LEVEL 1 RETRIEVAL POINTERS - 4 BYTES EACH
                                0735  1688 ;       BYTE 0 = BITS 16 - 23 OF LOGICAL BLOCK NUMBER
                                0735  1689 ;       BYTE 1 = COUNT - 1 OF BLOCKS COVERED BY THIS POINTER
                                0735  1690 ;       BYTES 2-3 = BITS 0:15 OF LOGICAL BLOCK NUMBER
                                0735  1691 ;
                      50   01 A4  9A  0735  1692          MOVZBL  1(R4),R0                ;COUNT - 1
                      51   02 A4  3C  0739  1693          MOVZWL  2(R4),R1                ;LOW LBN BITS
                  08  10   84  F0  073D  1694          INSV    (R4)+,#16,#8,R1         ;PUT HIGH LBN BITS IN
                           51         0741
                           36  11     0742  1695          BRB     INCRSB                  ;INCREMENT COUNT AND EXIT
                                0744  1696 ;
                                0744  1697 ; STRUCTURE LEVEL 2 RETRIEVAL POINTERS
                                0744  1698 ;       BITS 14:15 = RETRIEVAL POINTER FORMAT
                                0744  1699 ;
                  64   02   0E  EF  0744  1700 20$:    EXTZV   #FM2$V_FORMAT,#FM2$S_FORMAT,(R4),R0 ;FORMAT TO R0
                           50         0748
                                0749  1701          CASE    R0,<-
                                0749  1702                  PLACEMENT,-             ;PLACEMENT FORMAT
                                0749  1703                  FORMAT1,-               ;FORMAT 1
                                0749  1704                  FORMAT2-                ;FORMAT 2
                                0749  1705                  >
                                0753  1706 ;
                                0753  1707 ; FORMAT 3 = 8 BYTES
                                0753  1708 ;
                                0753  1709 ;       BITS 0:13 = BITS 16:29 OF COUNT - 1
                                0753  1710 ;       BITS 14:15 = FORMAT = 3
                                0753  1711 ;       BYTES 2-3 = BITS 0:15 OF COUNT - 1
                                0753  1712 ;       BYTES 4-7 = LOGICAL BLOCK NUMBER
                                0753  1713 ;
                                0753  1714 FORMAT3:
```

```
                50   84   10   9C  0753  1715          ROTL    #16,(R4)+,R0            ;FORM COUNT - 1
                02   1E   00   F0  0757  1716          INSV    #0,#30,#2,R0            ;ZERO HIGH 2 BITS
                          50        075B
                     51   84   D0  075C  1717          MOVL    (R4)+,R1                ;GET LBN
                          19   11  075F  1718          BRB     INCRSB                  ;INCREMENT COUNT AND EXIT
                                   0761  1719 ;
                                   0761  1720 ; PLACEMENT CONTROL - THIS IS NOT A RETRIEVAL POINTER, RATHER IT
                                   0761  1721 ;       CONSISTS OF 2 BYTES OF PLACEMENT INFORMATION.  TREAT AS IF 0
                                   0761  1722 ;       LENGTH RETRIEVAL POINTER.
                                   0761  1723 ;       R0 = 0
                                   0761  1724 ;
                                   0761  1725 PLACEMENT:
                     51   01   CE  0761  1726          MNEGL   #1,R1                   ;IMPOSSIBLE LBN
                     54   02   C0  0764  1727          ADDL    #2,R4                   ;BUMP THE POINTER
                          50   D4  0767  1728          CLRL    R0                      ;CLEAR BLOCK COUNT
                               05  0769  1729          RSB
                                   076A  1730 ;
                                   076A  1731 ; FORMAT 1 = 4 BYTES
                                   076A  1732 ;       BITS 0:7 = COUNT - 1
                                   076A  1733 ;       BITS 8:13 = BITS 16:21 OF LOGICAL BLOCK NUMBER
                                   076A  1734 ;       BYTES 2-3 = BITS 0:15  OF LOGICAL BLOCK NUMBER
                                   076A  1735 ;
                                   076A  1736 FORMAT1:
                     50   84   D0  076A  1737          MOVL    (R4)+,R0                ;FETCH ENTIRE RETRIEVAL POINTER
                50   06   08   EF  076D  1738          EXTZV   #FM2$V_HIGHLBN,#FM2$S_HIGHLBN,R0,R1 ;FETCH HIGH LBN BITS
                          51        0771
                50   50   10   79  0772  1739          ASHQ    #16,R0,R0               ;FORM R1 = LBN
                  50   FC   A4   9A  0776  1740          MOVZBL  -4(R4),R0               ;REFETCH COUNT - 1
                                   077A  1741 INCRSB:
                          50   D6  077A  1742          INCL    R0                      ;FORM COUNT
                               05  077C  1743          RSB                             ;AND RETURN
                                   077D  1744 ;
                                   077D  1745 ; FORMAT 2 = 6 BYTES
                                   077D  1746 ;
                                   077D  1747 ;       BITS 0:13 = COUNT - 1
                                   077D  1748 ;       BITS 14:15 = FORMAT = 2
                                   077D  1749 ;       BYTES 2-5 = LBN
                                   077D  1750 ;
                                   077D  1751 FORMAT2:
                     50   84   3C  077D  1752          MOVZWL  (R4)+,R0                ;FETCH COUNT - 1 AND FORMAT BITS
                50   0E   00   EF  0780  1753          EXTZV   #FM2$V_COUNT2,#FM2$S_COUNT2,R0,R0 ;COUNT - 1
                          50        0784
                     51   84   D0  0785  1754          MOVL    (R4)+,R1                ;LBN
                          F0   11  0788  1755          BRB     INCRSB                  ;INCREMENT COUNT AND RETURN
```

```
                        .SBTTL  STATBLK - GET FILE STATISTICS BLOCK
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       GIVEN A FILE HEADER, RETURN THE FILE STATISTICS BLOCK
                ;       AND OPTIONALLY RETURN THE RETRIEVAL POINTERS
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLG   ARGLIST,FIL$STATBLK
                ;
                ; INPUT PARAMETERS:
                ;
                ;       FILHDR(AP)      =               ;ADDRESS OF THE FILE HEADER
                ;       STATBLK(AP)     =               ;ADDRESS TO RETURN STATISTICS BLOCK
                ;       RTRVPTRLEN(AP)  =               ;ADDRESS TO RETURN THE NUMBER OF
                ;                                       ;BYTES OF RETRIEVAL POINTERS
                ;                                       ;FOUND IN THE FILE HEADER(S).
                ;                                       ;***** OPTIONAL PARAMETER *****
                ;       RTRVPTRBUF(AP)  =               ;ADDRESS OF RETRIEVAL POINTER
                ;                                       ;BUFFER DESCRIPTOR.  THIS PARAMETER
                ;                                       ;IS PRESENT IF AND ONLY IF
                ;                                       ;RTRVPTRLEN IS PRESENT.
                ;                                       ;ZERO DESCRIPTOR ADDRESS OR ZERO
                ;                                       ;BUFFER ADDRESS MEANS DON'T
                ;                                       ;RETURN RETRIEVAL POINTER INFO
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R0 = SYSTEM STATUS CODE
                ;       STATBLK CONTAINS 2 LONGWORDS
                ;               LBN OF 1ST BLOCK IF CONTIGUOUS OR ZERO IF NOT
                ;               SIZE OF FILE IN BLOCKS
                ;       RTRVPTRLEN RECEIVES THE NUMBER OF BYTES OF RETRIEVAL POINTER
                ;               INFORMATION THAT WOULD HAVE BEEN STORED IN THE RETRIEVAL
                ;               POINTER BUFFER GIVEN A LARGE ENOUGH BUFFER.
                ;       THE RETRIEVAL POINTER BUFFER RECEIVES NORMALIZED RETRIEVAL
                ;               POINTERS IN THE FORMAT 32 BIT COUNT, 32 BIT STARTING LBN
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
                ;       SS$_NORMAL                      SUCCESSFUL COMPLETION
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ; EQUATED SYMBOLS:
```

```
                                078A  1814 ;
                                078A  1815 ;    OFFSETS FROM AP
                                078A  1816 ;
                    00000000    078A  1817          ARGCNT          =       0       ;NUMBER OF ARGUMENTS
                    00000004    078A  1818          FILHDR          =       4       ;ADDRESS OF FILE HEADER
                    00000008    078A  1819          STATBLK         =       8       ;ADDRESS TO RETURN STATISTICS BLOCK
                    0000000C    078A  1820          RTRVPTRLEN      =       12      ;ADDRESS TO RETURN COUNT OF BYTES
                                078A  1821                                          ;STORED IN THE RETRIEVAL POINTER BUFFER
                    00000010    078A  1822          RTRVPTRBUF      =       16      ;ADDRESS OF RETRIEVAL POINTER
                                078A  1823                                          ;BUFFER DESCRIPTOR
                                078A  1824 ;
                                078A  1825 ;--
                                078A  1826
                                078A  1827 FIL$STATBLK::
                          00FC  078A  1828          .WORD   ^M<R2,R3,R4,R5,R6,R7>
                       56   7C  078C  1829          CLRQ    R6                      ;ASSUME NOT DOING RETRIEVAL POINTERS
                    04 6C   D1  078E  1830          CMPL    ARGCNT(AP),#RTRVPTRBUF/4 ;RTRV PTR PARAMS PRESENT?
                       0F   19  0791  1831          BLSS    5$                      ;BRANCH IF NOT
                 50   10 AC   D0  0793  1832          MOVL    RTRVPTRBUF(AP),R0       ;ADDRESS OF BUFFER DESCRIPTOR
                       09   13  0797  1833          BEQL    5$                      ;BRANCH IF NOT SPECIFIED
                    56 60   7D  0799  1834          MOVQ    (R0),R6                 ;R6 = MAX SIZE, R7 = BUFFER ADR
                    56 07   CA  079C  1835          BICL    #7,R6                   ;EVEN MULTIPLE OF 8 BYTES
                     0C BC  D4  079F  1836          CLRL    @RTRVPTRLEN(AP)         ;INIT RETURN BYTE COUNT
                 55  04 AC   D0  07A2  1837 5$:      MOVL    FILHDR(AP),R5           ;ADDRESS OF FILE HEADER
                 52  06 A5   3C  07A6  1838          MOVZWL  FH1$W_STRUCLEV(R5),R2   ;STRUCTURE LEVEL
                 50  34 A5   9A  07AA  1839          MOVZBL  FH2$L_FILECHAR(R5),R0   ;FILE CHARACTERISTICS IF LEVEL 2
                 04 52   09   E0  07AE  1840          BBS     #FH2$V_LEVEL2,R2,10$    ;BRANCH IF STRUCTURE LEVEL 2
                 50  0C A5   9A  07B2  1841          MOVZBL  FH1$W_FILECHAR(R5),R0   ;GET LEVEL 1 FILE CHARACTERISTICS
              50   01   07   EF  07B6  1842 10$:     EXTZV   #FH1$V_CONTIG,#1,R0,-(SP) ;CONTIGUOUS BIT TO TOP OF STACK
                       7E       07BA
                    FF55   30   07BB  1843          BSBW    INIRTRVPTRSCAN          ;INIT FOR SCAN OF RETRIEVAL POINTERS
                       53   D4  07BE  1844          CLRL    R3                      ;INIT REGISTER TO COUNT BLOCKS
                       29   11  07C0  1845          BRB     50$                     ;START AT BOTTOM OF LOOP IN CASE
                                07C2  1846                                          ;FILE HAS NO RETRIEVAL POINTERS
                    FF6C   30   07C2  1847 20$:     BSBW    GETRTRVPTR              ;GET THE NEXT RETRIEVAL POINTER
                       53   D5  07C5  1848          TSTL    R3                      ;IS THIS FIRST RTRV PTR?
                       0A   12  07C7  1849          BNEQ    40$                     ;BRANCH IF ALREADY COUNTED SOME
                                07C9  1850 ;
                                07C9  1851 ; FIRST RETRIEVAL POINTER
                                07C9  1852 ;
                       50   D5  07C9  1853          TSTL    R0                      ;IGNORE EMPTY ONES
                       1E   13  07CB  1854          BEQL    50$
                    03 6E   E9  07CD  1855          BLBC    (SP),40$                ;BRANCH IF FILE NOT CONTIGUOUS
                                07D0  1856                                          ;0(SP) = 0 IN THIS CASE
                 6E     51   D0  07D0  1857          MOVL    R1,(SP)                 ;SET LBN OF 1ST NON-ZERO RTRV PTR
                 53     50   C0  07D3  1858 40$:     ADDL    R0,R3                   ;ACCUMULATE COUNT OF BLOCKS
                        56   D5  07D6  1859          TSTL    R6                      ;ANY MORE ROOM FOR RTRV PTRS?
                        09   13  07D8  1860          BEQL    45$                     ;BRANCH IF NO MORE BUFFER SPACE
                 87     50   D0  07DA  1861          MOVL    R0,(R7)+                ;STORE SIZE OF RETRIEVAL POINTER
                 87     51   D0  07DD  1862          MOVL    R1,(R7)+                ;AND STORE LBN
                 56     08   C2  07E0  1863          SUBL    #8,R6                   ;USED 8 MORE BYTES OF SPACE
                        57   D5  07E3  1864 45$:     TSTL    R7                      ;DOES CALLER WANT RTRV PTR INFO?
                        04   13  07E5  1865          BEQL    50$                     ;BRANCH IF NOT, DON'T COUNT POINTERS
                0C BC   08   C0  07E7  1866          ADDL    #8,@RTRVPTRLEN(AP)      ;UPDATE RTRV PTR BYTE COUNT
                 55     54   D1  07EB  1867 50$:     CMPL    R4,R5                   ;ANY MORE RETRIEVAL POINTERS?
                        D2   1F  07EE  1868          BLSSU   20$                     ;BRANCH IF YES
                        04   BA  07F0  1869          POPR    #^M<R2>                 ;GET SAVED STARTING LBN
```

```
                08 BC    52    7D  07F2  1870          MOVQ    R2,@STATBLK(AP)         ;RETURN THE STATISTICS BLOCK
                   50    01    3C  07F6  1871          MOVZWL  #SS$_NORMAL,R0          ;SUCCESSFUL COMPLETION
                                 04  07F9  1872          RET
```

```
                                07FA  1874          .SBTTL  FIL$CHKFILHDR - CHECK FILE HEADER VALIDITY
                                07FA  1875 ;++
                                07FA  1876 ; FUNCTIONAL DESCRIPTION:
                                07FA  1877 ;
                                07FA  1878 ;       CHECK THE VALIDITY OF A FILE HEADER
                                07FA  1879 ;
                                07FA  1880 ; CALLING SEQUENCE:
                                07FA  1881 ;
                                07FA  1882 ;       BSBW    FIL$CHKFILHDR
                                07FA  1883 ;
                                07FA  1884 ; INPUT PARAMETERS:
                                07FA  1885 ;
                                07FA  1886 ;       R0 = ADDRESS OF FILE ID
                                07FA  1887 ;       R1 = ADDRESS OF FILE HEADER
                                07FA  1888 ;
                                07FA  1889 ; IMPLICIT INPUTS:
                                07FA  1890 ;
                                07FA  1891 ;       NONE
                                07FA  1892 ;
                                07FA  1893 ; OUTPUT PARAMETERS:
                                07FA  1894 ;
                                07FA  1895 ;       RSB TO CALLER IF FILE HEADER VALID
                                07FA  1896 ;       RET IF NOT VALID WITH R0 = ERROR STATUS
                                07FA  1897 ;
                                07FA  1898 ; IMPLICIT OUTPUTS:
                                07FA  1899 ;
                                07FA  1900 ;       NONE
                                07FA  1901 ;
                                07FA  1902 ; COMPLETION CODES:
                                07FA  1903 ;
                                07FA  1904 ;       SS$_BADFILEHDR          FILE ID CODES DON'T MATCH
                                07FA  1905 ;       SS$_NOSUCHFILE          FILE IS MARKED AS DELETED
                                07FA  1906 ;
                                07FA  1907 ; SIDE EFFECTS:
                                07FA  1908 ;
                                07FA  1909 ;       NONE
                                07FA  1910 ;
                                07FA  1911 ;--
                                07FA  1912
                                07FA  1913 FIL$CHKFILHDR:
                                07FA  1914          CASE    FH1$W_STRUCLEV+1(R1),<-
                                07FA  1915                  5$,-                    ;STRUCTURE LEVEL 1
                                07FA  1916                  10$-                    ;STRUCTURE LEVEL 2
                                07FA  1917                  >,TYPE=B,LIMIT=#1
                          27    11  0803  1918          BRB     30$                 ;BAD FILE HEADER
                                0805  1919 ;
                                0805  1920 ; STRUCTURE LEVEL 1
                                0805  1921 ;
                       7E   01  CE  0805  1922 5$:     MNEGL   #1,-(SP)            ;NO RVN TO CHECK
                    7E    02 A1  D0  0808  1923          MOVL    FH1$W_FID_NUM(R1),-(SP) ;PUSH FILE ID ON STACK
                          08    11  080C  1924          BRB     15$
                                080E  1925 ;
                                080E  1926 ; STRUCTURE LEVEL 2
                                080E  1927 ;
                    7E    0C A1  3C  080E  1928 10$:     MOVZWL  FH2$W_FID_RVN(R1),-(SP) ;PUSH RELATIVE VOLUME NUMBER
                    7E    08 A1  D0  0812  1929          MOVL    FH2$W_FID_NUM(R1),-(SP) ;PUSH FILE ID ON STACK
                          6E    B5  0816  1930 15$:     TSTW    (SP)                ;FILE DELETED?
```

```
                    18   13   0818  1931          BEQL    40$                     ;BRANCH IF YES
                8E  80   D1   081A  1932          CMPL    (R0)+,(SP)+             ;FILE NUM AND FILE SEQ NUM AGREE?
                    0D   12   081D  1933          BNEQ    30$                     ;BRANCH IF NOT, BAD HEADER
                    6E   D5   081F  1934          TSTL    (SP)                    ;CHECKING RVN?
                    05   19   0821  1935          BLSS    20$                     ;BRANCH IF NOT
                6E  60   B1   0823  1936          CMPW    (R0),(SP)               ;RELATIVE VOLUME NUMBER AND
                                0826  1937                                        ;FILE NUMBER EXTENSION AGREE
                    04   12   0826  1938          BNEQ    30$                     ;BRANCH IF NOT
                    01   BA   0828  1939 20$:     POPR    #^M<R0>                 ;CLEAN OFF STACK
                    0C   11   082A  1940          BRB     FIL$CHECKSUM            ;GO VERIFY THE CHECKSUM
          50  0810  8F   3C   082C  1941 30$:     MOVZWL  #SS$_BADFILEHDR,R0      ;THIS HEADER IS BAD
                         04   0831  1942          RET
          50  0910  8F   3C   0832  1943 40$:     MOVZWL  #SS$_NOSUCHFILE,R0      ;DELETED FILE
                         04   0837  1944          RET
```

```
                     0838  1946          .SBTTL  CHECKSUM - VALIDATE A CHECKSUM
                     0838  1947 ;++
                     0838  1948 ; FUNCTIONAL DESCRIPTION:
                     0838  1949 ;
                     0838  1950 ;       THIS ROUTINE CALCULATES AND CHECKS THE FILE11 CHECKSUM FOR
                     0838  1951 ; FILE HEADERS AND THE HOMEBLOCK.
                     0838  1952 ;
                     0838  1953 ; CALLING SEQUENCE:
                     0838  1954 ;
                     0838  1955 ;       BSBW    FIL$CHECKSUM            ;CHECK FILE HEADER CHECKSUM
                     0838  1956 ;       BSBW    FIL$CHECKSUM1           ;CHECK SPECIFIED NO. OF WORDS IN R0
                     0838  1957 ;
                     0838  1958 ; INPUT PARAMETERS:
                     0838  1959 ;
                     0838  1960 ;       R0 = NO. OF WORDS TO CHECK IF ENTERING AT CHECKSUM1
                     0838  1961 ;       R1 = ADDRESS OF BUFFER TO CHECK
                     0838  1962 ;
                     0838  1963 ; IMPLICIT INPUTS:
                     0838  1964 ;
                     0838  1965 ;       NONE
                     0838  1966 ;
                     0838  1967 ; OUTPUT PARAMETERS:
                     0838  1968 ;
                     0838  1969 ;       RSB TO CALLER IF CHECKSUM IS OK
                     0838  1970 ;       RET TO TOP LEVEL WITH ERROR CODE IN R0 IF CHECKSUM IS WRONG
                     0838  1971 ;
                     0838  1972 ; IMPLICIT OUTPUTS:
                     0838  1973 ;
                     0838  1974 ;       NONE
                     0838  1975 ;
                     0838  1976 ; COMPLETION CODES:
                     0838  1977 ;
                     0838  1978 ;       NONE
                     0838  1979 ;
                     0838  1980 ; SIDE EFFECTS:
                     0838  1981 ;
                     0838  1982 ;       NONE
                     0838  1983 ;
                     0838  1984 ;--
                     0838  1985
                     0838  1986 FIL$CHECKSUM:
50   00FF 8F    3C   0838  1987          MOVZWL  #FH1$W_CHECKSUM@-1,R0   ;NO. OF WORDS TO CHECK
                     083D  1988 FIL$CHECKSUM1:
          52    D4   083D  1989          CLRL    R2                      ;INIT THE SUM
                     083F  1990 10$:
     52   81    A0   083F  1991          ADDW    (R1)+,R2                ;ACCUMULATE THE SUM
       FA 50    F5   0842  1992          SOBGTR  R0,10$                  ;ONCE FOR EACH WORD
     61   52    B1   0845  1993          CMPW    R2,(R1)                 ;CHECKSUM OK?
          01    12   0848  1994          BNEQ    20$                     ;BRANCH IF NOT
                05   084A  1995          RSB
                     084B  1996 20$:
50   0808 8F    3C   084B  1997          MOVZWL  #SS$_BADCHKSUM,R0       ;ERROR STATUS IN R0
                04   0850  1998          RET
                     0851  1999
                     0851  2000
                     0851  2001          .END
```

```
ARGCNT                  = 00000000     D
ASSIGN_DEV                00001A5 R   D  02
BADDIR                    00000453 R   D  02
BADDIR1                   00000538 R   D  02
BADDIR2                   000003B4 R   D  02
BADFILNAM                 00000459 R   D  02
BADRET                    00000458 R   D  02
BADRET1                   000003B7 R   D  02
BUFADR                  = 0000000C     D
CACHE_ADR               = 00000010     D
CACHE_SIZE              = 0000000C     D
CHAN                    = 00000004     D
CHANADR                 = 00000004     D
DIR$B_NAMECOUNT         = 00000005     D
DIR$C_VERSION           = 00000008     D
DIR$T_NAME              = 00000006     D
DIR$W_FID               = 00000002     D
DIR$W_SIZE              = 00000000     D
DIR$W_VERSION           = 00000000     D
DIR...                  = FFFFFFFF     D
DIRBUF                  = 00000010     D
DIRNAM                    FFFFFFCC     D
DIR_BFCNT                 FFFFFFFC     D
DIR_BUF                   FFFFFFF8     D
DIR_CACHE_CNT           = 00000014     D
ENTRY                     FFFFFFD0     D
ENTRY_ADR                 FFFFFFF4     D
EXIT_FILID_FND            00000504 R   D  02
FAT$B_RATTRIB           = 00000001     D
FAT$B_RTYPE             = 00000000     D
FAT$C_FIXED             = 00000001     D
FAT$C_VARIABLE          = 00000002     D
FAT$L_EFBLK             = 00000008     D
FAT$M_NOSPAN            = 00000008     D
FAT$W_FFBYTE            = 0000000C     D
FAT$W_RSIZE             = 00000002     D
FH1$B_MPOFFSET          = 00000001     D
FH1$C_LEVEL1            = 00000101     D
FH1$V_CONTIG            = 00000007     D
FH1$V_LEVEL1            = 00000008     D
FH1$W_CHECKSUM          = 000001FE     D
FH1$W_FID_NUM           = 00000002     D
FH1$W_FILECHAR          = 0000000C     D
FH1$W_RECATTR           = 0000000E     D
FH1$W_STRUCLEV          = 00000006     D
FH1$W_VBNOFFSET         = 000001FE     D
FH2$B_MAP_INUSE         = 0000003A     D
FH2$B_MPOFFSET          = 00000001     D
FH2$C_LEVEL2            = 00000200     D
FH2$L_FILECHAR          = 00000034     D
FH2$V_BIGFILNUM         = 0000000A     D
FH2$V_CONTIG            = 00000007     D
FH2$V_DIRECTORY         = 0000000D     D
FH2$V_LEVEL2            = 00000009     D
FH2$W_CHECKSUM          = 000001FE     D
FH2$W_EXT_FID           = 0000000E     D
FH2$W_FID_NUM           = 00000008     D
FH2$W_FID_RVN           = 0000000C     D
FH2$W_RECATTR           = 00000014     D
FH2$W_STRUCLEV          = 00000006     D
FID                       FFFFFFFA     D
FID$B_NMX               = 00000005     D
FID$C_MFD               = 00000004     D
FIL$A_DIR_FID             00000000     D
FIL$A_DIR_OFID            0000001E     D
FIL$A_IXFHDR              00000018     D
FIL$B_DIR_LVL             00000012     D
FIL$CACHE_INIT            00000112 RG  D  02
FIL$CACHE_TRUNC           00000186 RG  D  02
FIL$CHECKSUM              00000838 R   D  02
FIL$CHECKSUM1             0000083D R   D  02
FIL$CHKFILHDR             000007FA R   D  02
FIL$CVTFILNAM             ********     X  02
FIL$C_CACHE_ID          = 00000001     D
FIL$C_DIR_SIZE            00000024   G D
FIL$C_SIZE                00000218   G D
FIL$FINDFILID             000002ED RG  D  02
FIL$GQ_CACHE              ********W GX   00
FIL$GT_DDDEV              ********W GX   00
FIL$GT_DDSTRING           ********   X   02
FIL$GT_TOPSYS             ********W GX   00
FIL$L_DIRMAX              0000000C     D
FIL$L_DIRNXT              00000008     D
FIL$L_DIROFF              00000004     D
FIL$L_DIR_BFOFF           00000018     D
FIL$L_DIR_LBN             00000014     D
FIL$L_LBNMAX              00000014     D
FIL$L_LBNNXT              00000010     D
FIL$L_LBNOFF              0000000C     D
FIL$MOUNT                 0000025D RG  D  02
FIL$OPENFILE              0000000C RG  D  02
FIL$Q_DIR_HDR             00000010     D
FIL$RDCHKFILHDR           000005F4 RG  D  02
FIL$RDWRTLBN              ********     X  02
FIL$READVBN               000006D3 RG  D  02
FIL$STATBLK               0000078A RG  D  02
FIL$T_DIR_NAM             00000006     D
FIL$WRITEVBN              000006CC RG  D  02
FIL$W_CACHE_ID            00000000     D
FIL$W_DIR_BFCNT           0000001C     D
FIL$W_DIR_BKCNT           00000010     D
FILDSC                  = 00000008     D
FILHDR                  = 00000004     D
FILID                   = 00000018     D
FILNAM                  = 00000008     D
FIL_GQ_CACHE              00000000 R   D  02
FIL_GT_DDDEV              00000004 R   D  02
FIL_GT_TOPSYS             00000008 R   D  02
FIND_LEVEL1               00000538 R   D  02
FIND_LEVEL1_1             00000546 R   D  02
FIND_LEVEL2               0000045F R   D  02
FIND_LEVEL2_1             0000046B R   D  02
FM1$B_INUSE             = 00000008     D
FM1$C_POINTERS          = 0000000A     D
```

```
FM1$W_EX_FILNUM                 =  00000002     D
FM2$S_COUNT2                    =  0000000E     D
FM2$S_FORMAT                    =  00000002     D
FM2$S_HIGHLBN                   =  00000006     D
FM2$V_COUNT2                    =  00000000     D
FM2$V_FORMAT                    =  0000000E     D
FM2$V_HIGHLBN                   =  00000008     D
FORMAT1                            0000076A R   D   02
FORMAT2                            0000077D R   D   02
FORMAT3                            00000753 R   D   02
FORMDIRSTRING                      000001F1 R   D   02
GETRTRVPTR                         00000731 R   D   02
HDRCNT                             FFFFFFFC     D
HM1$L_IBMAPLBN                  =  00000002     D
HM1$W_CHECKSUM1                 =  0000C03A     D
HM1$W_CHECKSUM2                 =  000001FE     D
HM1$W_IBMAPSIZE                 =  00000000     D
HM1$W_STRUCLEV                  =  0000000C     D
HM2$L_IBMAPLBN                  =  00000018     D
HM2$L_MAXFILES                  =  0000001C     D
HM2$W_CHECKSUM1                 =  0000003A     D
HM2$W_CHECKSUM2                 =  000001FE     D
HM2$W_CLUSTER                   =  0000000E     D
HM2$W_IBMAPSIZE                 =  00000020     D
HM2$W_STRUCLEV                  =  0000000C     D
INCRSB                             0000077A R   D   02
INIRTRVPTRSCAN                     00000713 R   D   02
IO$_READLBLK                    =  00000021     D
IO$_WRITELBLK                   =  00000020     D
IOFUNCTION                      =  FFFFFFFC     D
IXFHDR                          =  0000000C     D
LBN_CACHE_CNT                   =  00000018     D
LIB$CVT_DTB                        ********   X   02
NAMBLK                             FFFFFFA4     D
NAMDSC                             FFFFFF98     D
OPENFILE_1                         0000000C R   D   02
OPENFILE_2                         0000000E R   D   02
PLACEMENT                          00000761 R   D   02
RDWRTVBN                           000006D8 R   D   02
READ_DIR_HEADER                    000003B8 R   D   02
READ_DIR_LBN                       000005C6 R   D   02
RTRVPTRBUF                      =  00000010     D
RTRVPTRLEN                      =  0000000C     D
SAVABS...                       =  FFFFFFF0     D
SCRATCHSIZE                        FFFFFF98     D
SCRATCH_SIZE                       FFFFFFD0     D
SS$_BADCHKSUM                   =  00000808     D
SS$_BADFILEHDR                  =  00000810     D
SS$_BADFILENAME                 =  00000818     D
SS$_BADIRECTORY                 =  00000828     D
SS$_ENDOFFILE                   =  00000870     D
SS$_FILESTRUCT                  =  000008C0     D
SS$_NORMAL                      =  00000001     D
SS$_NOSUCHFILE                  =  00000910     D
STATBLK                         =  00000008     D
STORE3DIGITS                       000001D8 R   D   02
TMPRTRVDSC                         FFFFFFF0     D
TMPRTRVLEN                         FFFFFFF8     D
VBN                             =  00000008     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	00000851 (2129.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...							
ARGCNT	=00000000	1817 (17)	#-1462	(13)	#-1830	(17)	#-413	(2)	#-459	(2)
ASSIGN_DEV	000001A5-R	629 (5)	#-544	(3)						
BADDIR	00000453-R	1128 (9)	#-1040	(9)	#-1139	(10)	#-1144	(10)	#-1253	(10)
BADDIR1	00000538-R	1252 (10)	#-1262	(11)	#-1264	(11)				
BADDIR2	000003B4-R	1039 (9)	#-1059	(9)						
BADFILNAM	00000459-R	1130 (9)	#-1174	(10)	#-384	(2)	#-674	(6)	#-723	(7)
BADRET	00000458-R	1129 (9)								
BADRET1	000003B7-R	1041 (9)	#-1048	(9)						
BUFADR	=0000000C	1580 (14)	#-1621	(14)						
CACHE_ADR	=00000010	516 (3)	524	(3)						
CACHE_SIZE	=0000000C	515 (3)	524	(3)	#-525	(3)				
CHAN	=00000004	1578 (14)	#-1094	(9)	#-1377	(12)	#-1490	(13)	#-1623	(14)
			#-808	(8)						
CHANADR	=00000004	513 (3)	#-419.1	(2)	#-548	(3)				
DIR$B_NAMECOUNT	=00000005		#-1187	(10)						
DIR$C_VERSION	=00000008		#-1215	(10)						
DIR$T_NAME	=00000006		1188	(10)	1205	(10)				
DIR$W_FID	=00000002		#-1223	(10)	1225	(10)				
DIR$W_SIZE	=00000000		#-1191	(10)	#-1193	(10)	#-1206	(10)		
DIR$W_VERSION	=00000000		#-1212	(10)						
DIR...	=FFFFFFFF	1453 (13)	1453	(13)	153	(1)	169	(1)	294	(2)
			943	(9)						
DIRBUF	=00000010	931 (9)	#-1049	(9)	#-1376	(12)				
DIRNAM	FFFFFFCC	294 (2)	#-450.2	(2)	728	(7)	732	(7)	736	(7)
			739	(7)	746	(7)				
DIR_BFCNT	FFFFFFFC	943 (9)	#-1033	(9)	#-1100	(9)	#-1367	(12)	#-1369	(12)
DIR_BUF	FFFFFFF8	943 (9)	#-1034	(9)	#-1101	(9)	#-1370	(12)	#-1371	(12)
DIR_CACHE_CNT	=00000014	517 (3)	#-532	(3)						
ENTRY	FFFFFFD0	943 (9)	#-1020	(9)	#-1023	(9)	#-1024	(9)	#-1031	(9)
			#-1032	(9)	#-1033	(9)	#-1034	(9)	#-1036	(9)
			#-1071	(9)	#-1072	(9)	#-1073	(9)	#-1093	(9)
			#-1102	(9)	#-1106	(9)	#-1107	(9)	#-1233	(10)
			#-1238	(10)	#-1239	(10)	1245	(10)	962	(9)
			#-976	(9)	977	(9)	983	(9)	#-994	(9)
ENTRY_ADR	FFFFFFF4	943 (9)	#-1005	(9)	#-1235	(10)	#-986	(9)	#-993	(9)
EXIT_FILID_FND	00000504-R	1230 (10)	#-1336	(11)						
FAT$B_RATTRIB	=00000001		1141	(10)						
FAT$B_RTYPE	=00000000		1141	(10)	#-1143	(10)	#-1261	(11)		
FAT$C_FIXED	=00000001		#-1261	(11)						
FAT$C_VARIABLE	=00000002		#-1142	(10)						
FAT$L_EFBLK	=00000008		#-1067	(9)						
FAT$M_NOSPAN	=00000008		#-1142	(10)						
FAT$W_FFBYTE	=0000000C		#-1068	(9)						
FAT$W_RSIZE	=00000002		#-1263	(11)						
FH1$B_MPOFFSET	=00000001		116	(1)	#-1516	(13)	#-1650	(15)		
FH1$C_LEVEL1	=00000101		128	(1)						
FH1$V_CONTIG	=00000007		123	(1)	#-1842	(17)				
FH1$V_LEVEL1	=00000008	129 (1)								
FH1$W_CHECKSUM	=000001FE		118	(1)	122	(1)	125	(1)	#-1987	(19)
FH1$W_FID_NUM	=00000002		#-1923	(18)						

```
FH1$W_FILECHAR              =0000000C                    #-1841   (17)
FH1$W_RECATTR               =0000000E                      1063   (9)
FH1$W_STRUCLEV              =00000006                    #-1064   (9)      117    (1)    #-1600   (14)   #-1838   (17)
                                                         #-1917   (18)
FH1$W_VBNOFFSET             =000001FE      125   (1)     #-1487   (13)   #-855   (8)
FH2$B_MAP_INUSE             =0000003A                    #-1657   (15)
FH2$B_MPOFFSET              =00000001                      116    (1)
FH2$C_LEVEL2                =00000200                      131    (1)
FH2$L_FILECHAR              =00000034                      1139   (10)   #-1839   (17)
FH2$V_BIGFILNUM             =0C0000A       133   (1)     #-1482   (13)   #-862   (8)
FH2$V_CONTIG                =00000007                      123    (1)
FH2$V_DIRECTORY             =0000000D                    #-1139   (10)
FH2$V_LEVEL2                =00000009      132   (1)     #-1515   (13)   #-1652   (15)   #-1685   (16)   #-1840   (17)
FH2$W_CHECKSUM              =000001FE                      118    (1)
FH2$W_EXT_FID               =0000C0E                     #-1514   (13)
FH2$W_FID_NUM               =00000008                    #-1929   (18)
FH2$W_FID_RVN               =0000000C                    #-1928   (18)
FH2$W_RECATTR               =00000014                      1066   (9)
FH2$W_STRUCLEV              =00000006                      117    (1)      1482   (13)     1515   (13)     862    (8)
FID                          FFFFFFFA      294   (2)       416    (2)    #-433   (2)    #-434   (2)
FID$B_NMX                   =00000005                    #-1483   (13)
FID$C_MFD                   =00000004                    #-433    (2)    #-434   (2)
FIL$A_DIR_FID                00000000      169   (1)       963    (9)      982   (9)      983   (9)      984   (9)
FIL$A_DIR_OFID               0000001E      169   (1)     #-1011   (9)      1013   (9)    #-1238   (10)   #-1239   (10)
FIL$A_IXFHDR                 00000018      153   (1)       424    (2)      546   (3)
FIL$B_DIR_LVL                00000012      169   (1)     #-1036   (9)    #-1073   (9)
FIL$CACHE_INIT               00000112-R    521   (3)
FIL$CACHE_TRUNC              00000186-R    596   (4)
FIL$CHECKSUM                 00000838-R   1986   (19)  #-1940   (18)   #-816   (8)
FIL$CHECKSUM1                000U083D-R   1988   (19)  #-814    (8)
FIL$CHKFILHDR                000007FA-R   1913   (18)  #-1495   (13)   #-853   (8)
FIL$CVTFILNAM                00000000-XR                  1274   (11)
FIL$C_CACHE_ID              =00000001      138   (1)     #-356    (2)    #-528   (3)
FIL$C_DIR_SIZE               00000024      169   (1)     #-1241   (10)   #-1245   (10)     176    (1)    #-532    (3)
                                                           943    (9)    #-996   (9)
FIL$C_SIZE                   00000218      153   (1)       176    (1)    #-526   (3)    #-530   (3)    #-531    (3)
                                                         #-535    (3)
FIL$FINDFILID                000002ED-R    947   (9)       449    (2)
FIL$GQ_CACHE                 00000000-XR                  180    (1)      190   (1)    #-354   (2)    #-551    (3)
                                                         #-598    (4)    #-601   (4)
FIL$GT_DDDEV                 00000000-XR                  181    (1)      192   (1)      633   (5)
FIL$GT_DDSTRING              00000000-XR                  359    (2)
FIL$GT_TOPSYS                00000000-XR                  182    (1)      194   (1)      437   (2)
FIL$L_DIRMAX                 0000000C      153   (1)     #-1242   (10)   #-535   (3)      537   (3)    #-599    (4)
FIL$L_DIRNXT                 00000008      153   (1)     #-1240   (10)   #-1244   (10)   #-531   (3)    #-599    (4)
                                                         #-979    (9)
FIL$L_DIROFF                 00000004      153   (1)     #-530    (3)    #-978   (9)
FIL$L_DIR_BFOFF              00000018      169   (1)     #-1023   (9)    #-1034   (9)    #-1107   (9)
FIL$L_DIR_LBN                00000014      169   (1)     #-1031   (9)    #-1071   (9)    #-1093   (9)
FIL$L_LBNMAX                 00000014      153   (1)     #-1079   (9)    #-543   (3)    #-600   (4)
FIL$L_LBNNXT                 00000010      153   (1)     #-1079   (9)    #-1086   (9)    #-1109   (9)    #-538    (3)
                                                         #-543    (3)    #-600   (4)    #-601   (4)
FIL$L_LBNOFF                 0000000C      153   (1)       537    (3)    #-538   (3)
FIL$MOUNT                    0000025D-R    801   (8)       426    (2)      549   (3)
FIL$OPENFILE                 0000000C-R    328   (2)
FIL$Q_DIR_HDR                00000010      169   (1)     #-1020   (9)
FIL$RDCHKFILHDR              000005F4-R   1457   (13)      1047   (9)      462   (2)
```

```
FIL$RDWRTLBN                    00000000-XR                      1095  (9)      1377  (12)     1624  (14)      810  (8)
                                                                 846  (8)
FIL$READVBN                     000006D3-R  1594  (14)         1490  (13)
FIL$STATBLK                     0000078A-R  1827  (17)         1501  (13)
FIL$T_DIR_NAM                   00000006    169   (1)       #-1102  (9)    #-1233  (10)   #-976  (9)       977  (9)
                                                               982  (9)    #-994  (9)
FIL$WRITEVBN                    000006CC-R  1589  (14)
FIL$W_CACHE_ID                  00000000    153   (1)       #-356  (2)     #-528  (3)
FIL$W_DIR_BFCNT                 0000001C    169   (1)       #-1024  (9)    #-1033  (9)    #-1106  (9)
FIL$W_DIR_BKCNT                 00000010    169   (1)       #-1032  (9)    #-1072  (9)
FILDSC                         =00000008    929   (9)       #-1152  (10)   #-1269  (11)   #-965  (9)
FILHDR                         =00000004    1818  (17)        1469  (13)   #-1599  (14)   #-1837  (17)   #-417  (2)
FILID                          =00000018    1443  (13)        1013  (9)      1225  (10)   #-1279  (11)   #-1318  (11)
                                                           #-1325  (11)   #-1333  (11)   #-1334  (11)      1472  (13)
                                                           #-1526  (13)     964  (9)
FILNAM                         =00000008    514   (3)       #-366  (2)     #-637  (5)
FIL_GQ_CACHE                    00000000-R  189   (1)       #-342  (2)
FIL_GT_DDDEV                    00000004-R  191   (1)       #-631  (5)
FIL_GT_TOPSYS                   00000008-R  193   (1)       #-435  (2)
FIND_LEVEL1                     0000053B-R  1260  (11)      #-1123  (9)
FIND_LEVEL1_1                   00000546-R  1268  (11)      #-1038  (9)
FIND_LEVEL2                     0000045F-R  1138  (10)      #-1122  (9)
FIND_LEVEL2_1                   0000046B-R  1151  (10)      #-1037  (9)
FM1$B_INUSE                    =00000008                    #-1653  (15)
FM1$C_POINTERS                 =0000000A                    #-1654  (15)
FM1$W_EX_FILNUM                =00000002                    #-1518  (13)
FM2$S_COUNT2                   =0000000E                    #-1753  (16)
FM2$S_FORMAT                   =00000002                    #-1700  (16)
FM2$S_HIGHLBN                  =00000006                    #-1738  (16)
FM2$V_COUNT2                   =00000000                    #-1753  (16)
FM2$V_FORMAT                   =0000000E                    #-1700  (16)
FM2$V_HIGHLBN                  =00000008                    #-1738  (16)
FORMAT1                         0000076A-R  1736  (16)        1705  (16)
FORMAT2                         0000077D-R  1751  (16)        1705  (16)
FORMAT3                         00000753-R  1714  (16)
FORMDIRSTRING                   000001F1-R  713   (7)       #-442  (2)     #-445  (2)
GETRTRVPTR                      00000731-R  1684  (16)      #-1608  (14)   #-1847  (17)
HDRCNT                          FFFFFFFC    1453  (13)      #-1489  (13)     1492  (13)   #-1524  (13)      1527  (13)
HM1$L_IBMAPLBN                 =00000002                    #-835  (8)
HM1$W_CHECKSUM1                =0000003A                      120  (1)     #-813  (8)
HM1$W_CHECKSUM2                =000001FE                      121  (1)       122  (1)
HM1$W_IBMAPSIZE                =00000000                    #-836  (8)
HM1$W_STRUCLEV                 =0000000C                      119  (1)     #-820  (8)
HM2$L_IBMAPLBN                 =00000018                    #-826  (8)
HM2$L_MAXFILES                 =0000001C                    #-828  (8)
HM2$W_CHECKSUM1                =0000003A                      120  (1)
HM2$W_CHECKSUM2                =000001FE                      121  (1)
HM2$W_CLUSTER                  =0000000E                    #-830  (8)
HM2$W_IBMAPSIZE                =00000020                    #-827  (8)
HM2$W_STRUCLEV                 =0000000C                      119  (1)
INCRSB                          0000077A-R  1741  (16)      #-1695  (16)   #-1718  (16)   #-1755  (16)
INIRTRVPTRSCAN                  00000713-R  1649  (15)      #-1601  (14)   #-1843  (17)
IO$_READLBLK                   =00000021                    #-1091  (9)    #-1377  (12)   #-1596  (14)   #-805  (8)
IO$_WRITELBLK                  =00000020                    #-1591  (14)
IOFUNCTION                     =FFFFFFFC    1585  (14)      #-1620  (14)
IXFHDR                         =0000000C    1440  (13)        1469  (13)   #-1470  (13)   #-1496  (13)   #-418  (2)
                                                           #-424  (2)     #-803  (8)
```

```
LBN_CACHE_CNT                   =00000018      518     (3)     #-539     (3)
LIB$CVT_DTB                      00000000-XR                     1173     (10)
NAMBLK                           FFFFFFA4      294     (2)       337     (2)     #-450.2   (2)
NAMDSC                           FFFFFF98      294     (2)     #-337     (2)       419     (2)     #-446     (2)     #-455     (2)
                                                               #-457     (2)
OPENFILE_1                       0000000C-R    333     (2)
OPENFILE_2                       0000000E-R    335     (2)
PLACEMENT                        00000761-R    1725    (16)      1705    (16)
RDWRTVBN                         000006D8-R    1598    (14)    #-1592    (14)
READ_DIR_HEADER                  000003B8-R    1046    (9)     #-1006    (9)
READ_DIR_LBN                     000005C6-R    1366    (12)    #-1198    (10)    #-1293    (11)
RTRVPTRBUF                      =00000010      1822    (17)    #-1462    (13)    #-1464    (13)    #-1830    (17)    #-1832    (17)
                                                               #-413     (2)     #-459     (2)
RTRVPTRLEN                      =0000000C      1820    (17)    #-1467    (13)    #-1504    (13)    #-1836    (17)    #-1866    (17)
                                                               #-415     (2)
SAVABS...                       =FFFFFFF0      1453    (13)
SCRATCHSIZE                      FFFFFF98      294     (2)     #-336     (2)
SCRATCH_SIZE                     FFFFFFD0      943     (9)     #-949     (9)     #-950     (9)       962     (9)
SS$_BADCHKSUM                   =00000808                      #-1997    (19)
SS$_BADFILEHDR                  =00000810                      #-1941    (18)
SS$_BADFILENAME                 =00000818                      #-1131    (9)
SS$_BADIRECTORY                 =00000828                      #-1128    (9)
SS$_ENDOFFILE                   =00000870                      #-1614    (14)
SS$_FILESTRUCT                  =000008C0                      #-821     (8)
SS$_NORMAL                      =00000001                      #-1014    (9)     #-1247    (10)    #-1528    (13)    #-1871    (17)
                                                               #-552     (3)     #-602     (4)     #-864     (8)
SS$_NOSUCHFILE                  =00000910                      #-1196    (10)    #-1328    (11)    #-1943    (18)
STATBLK                         =00000008      1819    (17)    #-1050    (9)       1472    (13)    #-1473    (13)    #-1870    (17)
STORE3DIGITS                     000001D8-R    671     (6)     #-729     (7)     #-733     (7)
TMPRTRVDSC                       FFFFFFF0      1453    (13)    #-1466    (13)      1497    (13)    #-1505    (13)    #-1507    (13)
                                                               #-1508    (13)    #-1509    (13)
TMPRTRVLEN                       FFFFFFF8      1453    (13)      1498    (13)    #-1502    (13)
VBN                             =00000008      1579    (14)    #-1607    (14)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1		77 (1) 78 (1) 79 (1) 80 (1) 81 (1) 82 (1) 83 (1) 84 (1) 85 (1) 86 (1) 87 (1) 88 (1)
$FATDEF	3	77 (1)	77 (1)
$FH1DEF	2	78 (1)	78 (1)
$FH2DEF	5	79 (1)	79 (1)
$FIDDEF	1	82 (1)	82 (1)
$FM1DEF	2	80 (1)	80 (1)
$FM2DEF	2	81 (1)	81 (1)
$HM1DEF	2	83 (1)	83 (1)
$HM2DEF	4	84 (1)	84 (1)
$IODEF	17	85 (1)	85 (1)
$OFFSET	2	142 (1)	142 (1) 1449 (13) 157 (1) 288 (2) 937 (9)
$OFFST1	1	153 (1)	1453 (13) 153 (1) 169 (1) 294 (2) 943 (9)
$PSLDEF	2	86 (1)	86 (1)
$SSDEF	21	87 (1)	87 (1)
$VADEF	1	88 (1)	88 (1)
ASSUME	1	116 (1)	1141 (10) 116 (1) 117 (1) 118 (1) 119 (1) 120 (1) 121 (1) 122 (1) 123 (1) 128 (1) 131 (1) 1469 (13) 1472 (13) 524 (3) 537 (3) 962 (9) 963 (9) 982 (9)
CASE	1	817 (8)	1701 (16) 1914 (18) 817 (8)
READLBN	1	102 (1)	1377 (12)
READVBN	1	92 (1)	1490 (13)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.10	00:00:00.26
Command processing	135	00:00:00.78	00:00:01.66
Pass 1	867	00:00:19.51	00:00:28.41
Symbol table sort	0	00:00:02.09	00:00:04.11
Pass 2	451	00:00:06.21	00:00:14.57
Symbol table output	5	00:00:00.15	00:00:00.20
Psect synopsis output	5	00:00:00.02	00:00:00.02
Cross-reference output	55	00:00:00.93	00:00:01.29
Assembler run totals	1558	00:00:29.81	00:00:50.52

The working set limit was 1000 pages.
99309 bytes (194 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1297 non-local and 111 local symbols.
2005 source lines were read in Pass 1, producing 0 object records in Pass 2.
76 pages of virtual memory were used to define 25 macros.

+---------------------------+
! Macro library statistics !
+---------------------------+

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	10
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	20

1360 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) FILEREAD/UPDA=(FILEREAD.UPD,FILEREAD.ENH)+SYS$LIBRARY:LIB/L.BRARY+DMA1:[SYSO
```

Table of contents

(1) 84 Libraries, Macros, Equated Symbols
(1) 114 Data Psect Declarations
(1) 136 VRSetFlg Routine - Set control flags
(1) 191 VRClrFlg Routine - Clear control flags
(1) 237 VRSetEF Routine - Set event flags
(1) 287 VRClrEF Routine - Clear event flags
(1) 337 VRShowFlg Routine - Show control flags
(1) 407 VRShowEF Routine - Show event flags

```
0000      1         .TITLE  FLAGS   *** FLAGS Set/Clear/Show flags
0000      2         .IDENT  /07-13/
0000      3         .NoShow Conditionals    ;                                                    [11]
0000      4
0000      5 ;++
0000      6 ; Copyright (c) 1977, 1982
0000      7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      8 ;
0000      9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     15 ; REMAIN IN DEC.
0000     16 ;
0000     17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     19 ; CORPORATION.
0000     20 ;
0000     21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     23 ;--
```

```
0000     25 ;++
0000     26 ; FACILITY:
0000     27 ;
0000     28 ;       VAX DIAGNOSTIC SUPERVISOR
0000     29 ;
0000     30 ; ABSTRACT:
0000     31 ;
0000     32 ;       Set/clear/show control and event flags.
0000     33 ;
0000     34 ; ENVIRONMENT:
0000     35 ;
0000     36 ; AUTHOR:
0000     37 ;
0000     38 ;       TOM SOUTTER     17-NOV-77       VERSION 01
0000     39 ;
0000     40 ; MODIFIED BY:
0000     41 ;       01      TOM SOUTTER     21-NOV-77       VERSION 02
0000     42 ;               DSPR #14 - DEFAULT FLAGS SETTING.
0000     43 ;
0000     44 ;       02      KEN CHAPMAN     05-DEC-77       VERSION 03 (ESSAA-3.05)
0000     45 ;               DSPR #45 - REMOVED HALTI.
0000     46 ;
0000     47 ;       03      KEN CHAPMAN     09-JAN-78       VERSION 04 (ESSAA-3.06)
0000     48 ;               DSPR #14 - DEFAULT FLAG SETTING (AND CLEARING).
0000     49 ;       04      SET AND CLEAR FLAGS ALL SHOULDN'T MODIFY APT FLAGS.
0000     50 ;
0000     51 ;       05      KEN CHAPMAN     19-JAN-78       VERSION 05 (ESSAA-3.07)
0000     52 ;               FIXED SET AND CLEAR FLAGS ALL BY DEFINING DSA$M_ALLFLAGS.
0000     53 ;
0000     54 ;       06      Dave Butenhof   15-feb-80
0000     55 ;               Include SEARCH flag
0000     56 ;
0000     57 ;       07      Dave Butenhof   30-apr-80
0000     58 ;               Change DSA$GL_FLAGS references from word relative to long
0000     59 ;               relative
0000     60 ;
0000     61 ;       08      - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     62 ;               Fixed truncation errors by changing remaining W^'s to L^.
0000     63 ;               Also added .LIBRARY statements for $DS and $DIAG.
0000     64 ;
0000     65 ;       09      - Jack Stansbury, 4-Nov-1981, Version 6.-
0000     66 ;               Took out all references to the following:
0000     67 ;               LOCK, SPOOL, COMPAT, ENSPEC, HALTD, HALTI, SUBMIT, LOG, and
0000     68 ;               CONSOLE. These flags are no longer used anywhere. I changed
0000     69 ;               the HALTD and HALTI flags to just plain HALT.
0000     70 ;
0000     71 ;       10      - Jack Stansbury, 12-Nov-1981, Version 6.-
0000     72 ;               Added the VERIFY and BINARY flags.
0000     73 ;
0000     74 ;       11      - Jack Stansbury, 25-Mar-1982, Version 6.?
0000     75 ;               Added typecoding to the print statements.
0000     76 ;
0000     77 ;       [12] Dave Butenhof, 07-May-1982, version 6.8
0000     78 ;               Fix truncation error
0000     79 ;
0000     80 ;       [13]    Marion Baggett, 16-June-1982, Version 6.8
0000     81 ;               Show HALT  flag if cleared.
```

```
                                0000     82 ;--
```

```
                        0000     84          .SBTTL  Libraries, Macros, Equated Symbols
                        0000     85 ;
                        0000     86 ; INCLUDE FILES:
                        0000     87 ;
                        0000     88          .Library        /Sys$Library:Lib/       ;                                       [11]
                        0000     89          .Library        /$DS/                   ;                                       [08]
                        0000     90          .Library        /$DIAG/                 ;                                       [08]
                        0000     91
                        0000     92 ;
                        0000     93 ; MACROS:
                        0000     94 ;
                        0000     95
                        0000     96 ;
                        0000     97 ; EQUATED SYMBOLS:
                        0000     98 ;
                        0000     99
                        0000    100          $DS_DSADEF                      ; CONTROL FLAG DEFINITIONS
                        0000    101          CLIDEF                          ; CLI COMMAND BLOCK OFFSETS
                        0000    102          ENVDEF                          ; ENVIRONMENT BIT DEFINITIONS
                        0000    103          $DS_TypeDef                     ; Define TypeCodes                              [11]
                        0000    104
 00000002               0000    105          $ENV            = 1@V_ENV       ; FORCE SS CALLS TO SS VECTORS
                        0000    106
 0000000E               0000    107          LEFTMOSTFLAG    = DSA$V_SEARCH  ; MSB for CVTREG function
                        0000    108
                        0000    109          DSA$M_ALLFLAGS  = DSA$M_HALT  ! DSA$M_LOOP    ! DSA$M_BELL  -   ;              [10]
                        0000    110                          ! DSA$M_IE1   ! DSA$M_IE2     ! DSA$M_IE3   -   ;              [09]
                        0000    111                          ! DSA$M_QUICK ! DSA$M_VERIFY  ! DSA$M_TRACE -   ;              [10]
 000037FD               0000    112                          ! DSA$M_OPER  ! DSA$M_PROMPT  ! DSA$M_IES     ;              [09]
```

```
                                        0000   114          .Subtitle       Data Psect Declarations
                                        0000   115 ;
                                        0000   116 ; OWN STORAGE:
                                        0000   117 ;
                                        0000   118
                                    00000000   119          .Psect  Data, Shr, NoExe, NoWrt, Byte
                                        0000   120
                                        0000   121 FMTSHOWFLGS:
66 20 6C 6F 72 74 6E 6F 43 2F 21 00'    0000   122          .ASCIC  .!/Control flags  set : !AC.
20 3A 20 74 65 73 20 20 73 67 61 6C     000C
                        43 41 21        0018
                              1A        0000
                                        001B   123
                                        001B   124 FMTSHOWFLGC:
66 20 6C 6F 72 74 6E 6F 43 2F 21 00'    C01B   125          .ASCIC  .!/Control flags clear: !AC!/.
20 3A 72 61 65 6C 63 20 73 67 61 6C     0027
                     2F 21 43 41 21     0033
                              1C        001B
                                        0038   126
                                        0038   127 FMTSHOWEF:
61 6C 66 20 74 6E 65 76 45 2F 21 00'    0038   128          .ASCIC  .!/Event flags set: !AC!/.
21 43 41 21 20 3A 74 65 73 20 73 67     0044
                                 2F     0050
                                 18     0038
                                        0051   129
                                        0051   130 AFLAGMNE:
6D 6F 72 50 2C 68 63 72 61 65 53 00'    0051   131          .ASCIC  .Search,Prompt,Oper,,Trace,Verify,Quick,,-      ;               [10]
61 72 54 2C 2C 72 65 70 4F 2C 74 70     005D
75 51 2C 79 66 69 72 65 56 2C 65 63     0069
                              2C 6B 63 69       0075
2C 32 45 49 2C 33 45 49 2C 53 45 49     0079   132                  .IES,IE3,IE2,IE1,Bell,Loop,Binary,Halt.         ;               [10]
6F 6F 4C 2C 6C 6C 65 42 2C 31 45 49     0085
6C 61 48 2C 79 72 61 6E 69 42 2C 70     0091
                                 74     009D
                                 4C     0051
                                        009E   133 AEFNMNE:
30 32 2C 31 32 2C 33 32 2C 33 32 00'    009E   134          .ASCIC  .23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1.
36 31 2C 37 31 2C 38 31 2C 39 31 2C     00AA
32 31 2C 33 31 2C 34 31 2C 35 31 2C     0086
37 2C 38 2C 39 2C 30 31 2C 31 31 2C     00C2
31 2C 32 2C 33 2C 34 2C 35 2C 36 2C     00CE
                                 38     009E
```

```
                     00DA   136          .SBTTL  VRSetFlg Routine - Set control flags
                  00000000  137          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   138 ;++
                     0000   139 ; FUNCTIONAL DESCRIPTION:
                     0000   140 ;
                     0000   141 ;       SETS SPECIFIED OPERATOR CONTROL FLAGS
                     0000   142 ;
                     0000   143 ; CALLING SEQUENCE:
                     0000   144 ;
                     0000   145 ;       BSBW    VRSETFLG
                     0000   146 ;
                     0000   147 ; INPUT PARAMETERS:
                     0000   148 ;
                     0000   149 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
                     0000   150 ;
                     0000   151 ; IMPLICIT INPUTS:
                     0000   152 ;
                     0000   153 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
                     0000   154 ;
                     0000   155 ; OUTPUT PARAMETERS:
                     0000   156 ;
                     0000   157 ;       NONE
                     0000   158 ;
                     0000   159 ; IMPLICIT OUTPUTS:
                     0000   160 ;
                     0000   161 ;       NONE
                     0000   162 ;
                     0000   163 ; COMPLETION CODES:
                     0000   164 ;
                     0000   165 ;       NONE
                     0000   166 ;
                     0000   167 ; SIDE EFFECTS:
                     0000   168 ;
                     0000   169 ;       NONE
                     0000   170 ;
                     0000   171 ;--
```

```
                          0000   173 VRSETFLG::                                ;
                   0B 62  0C  E1  0000   174          BBC     #CLI$V_DEFAULT, -        ; SET TO DEFAULT ?
                          0004   175                    CLI$L_FLAGS(R2), 10$
         0000FE00'EF  0000'8F  B0  0004   176          MOVW    #DSA$M_DFLTFLGS, -       ; Yes, plug in default
                          000D   177                    L^DSA$GL_FLAGS
                       13  11  000D   178          BRB     VRSETFLG_X
                          000F   179
                    1C A2  D5  000F   180 10$:     TSTL    CLI$L_DATA(R2)           ; CHECK FOR ALL.
                       06  18  0012   181          BGEQ    20$                      ; SKIP IF NOT.
              1C A2  37FD 8F  3C  0014   182          MOVZWL  #DSA$M_ALLFLAGS, -       ; GET "ALL" FLAGS.
                          001A   183                    CLI$L_DATA(R2)
                          001A   184
         0000FE00'EF  1C A2  C8  001A   185 20$:     BISL2   CLI$L_DATA(R2), -        ; No, set the specified flags
                          0022   186                    L^DSA$GL_FLAGS
                          C022   187
                          0022   188 VRSETFLG_X:
                       05  0022   189          RSB                              ; RETURN
```

```
0023   191          .SBTTL  VRClrFlg Routine - Clear control flags
0023   192 ;++
0023   193 ; FUNCTIONAL DESCRIPTION:
0023   194 ;
0023   195 ;       CLEARS SPECIFIED OPERATOR CONTROL FLAGS
0023   196 ;
0023   197 ; CALLING SEQUENCE:
0023   198 ;
0023   199 ;       BSBW    VRCLRFLG
0023   200 ;
0023   201 ; INPUT PARAMETERS:
0023   202 ;
0023   203 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
0023   204 ;
0023   205 ; IMPLICIT INPUTS:
0023   206 ;
0023   207 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
0023   208 ;
0023   209 ; OUTPUT PARAMETERS:
0023   210 ;
0023   211 ;       NONE
0023   212 ;
0023   213 ; IMPLICIT OUTPUTS:
0023   214 ;
0023   215 ;       NONE
0023   216 ;
0023   217 ; COMPLETION CODES:
0023   218 ;
0023   219 ;       NONE
0023   220 ;
0023   221 ; SIDE EFFECTS:
0023   222 ;
0023   223 ;       NONE
0023   224 ;
0023   225 ;--
```

```
                                0023   227 VRCLRFLG::                                  ;
                      1C A2     D5   0023   228          TSTL    CLI$L_DATA(R2)         ; CHECK FOR ''ALL''.
                         06     18   0026   229          BGEQ    10$                    ; BR IF NOT.
              1C A2   37FD 8F    3C   0028   230          MOVZWL  #DSA$M_ALLFLAGS, -     ; GET ''ALL'' FLAGS.
                                0028   231                    CLI$L_DATA(R2)
                                002E   232
     0000FE00'EF      1C A2     CA   002E   233 10$:     BICL2   CLI$L_DATA(R2), -      ; Clear specified flags
                                0036   234                    L^DSA$GL_FLAGS
                                   05   0036   235          RSB                        ; RETURN
```

```
0037   237          .SBTTL  VRSetEF Routine - Set event flags
0037   238 ;++
0037   239 ; FUNCTIONAL DESCRIPTION:
0037   240 ;
0037   241 ;       SETS OPERATOR SPECIFIED EVENT FLAGS (1-23 ONLY)
0037   242 ;
0037   243 ; CALLING SEQUENCE:
0037   244 ;
0037   245 ;       BSBW    VRSETEF
0037   246 ;
0037   247 ; INPUT PARAMETERS:
0037   248 ;
0037   249 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
0037   250 ;
0037   251 ; IMPLICIT INPUTS:
0037   252 ;
0037   253 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
0037   254 ;
0037   255 ; OUTPUT PARAMETERS:
0037   256 ;
0037   257 ;       NONE
0037   258 ;
0037   259 ; IMPLICIT OUTPUTS:
0037   260 ;
0037   261 ;       NONE
0037   262 ;
0037   263 ; COMPLETION CODES:
0037   264 ;
0037   265 ;       NONE
0037   266 ;
0037   267 ; SIDE EFFECTS:
0037   268 ;
0037   269 ;       NONE
0037   270 ;
0037   271 ;--
```

```
                                0037   273 VRSETEF::                                ;
                          08    0037   274          PUSHR   #^M<R3>                 ; SAVE SOME WORKING REGISTERS
      1C A2   FF000001    8F    0039   275          BICL    #^XFF000001, -          ; CLEAR ILLEGAL FLAGS
                                0041   276            CLI$L_DATA(R2)
                                0041   277
                    53    17    0041   278 10$:     MOVL    #23,R3                  ; UPPER FLAG NUMBER LIMIT
                                0044   279
            09 1C A2    53    E1    0044   280 20$:     BBC     R3, CLI$L_DATA(R2), 30$ ; EFN SPECIFIED ?
                                0049   281          $SETEF_S  R3                    ; YES, THEN GO SET IT
                                0052   282
                    EF  53    F5    0052   283 30$:     SOBGTR  R3,20$                  ; NEXT TILL DONE (DON'T DO 0)
                          08    BA    0055   284          POPR    #^M<R3>             ; RESTORE REGISTERS
                                05    0057   285          RSB                         ; RETURN
```

```
0058    287          .SBTTL  VRClrEF Routine - Clear event flags
0058    288 ;++
0058    289 ; FUNCTIONAL DESCRIPTION:
0058    290 ;
0058    291 ;       CLEARS OPERATOR SPECIFIED EVENT FLAGS (1-23 ONLY)
0058    292 ;
0058    293 ; CALLING SEQUENCE:
0058    294 ;
0058    295 ;       BSBW    VRCLREF
0058    296 ;
0058    297 ; INPUT PARAMETERS:
0058    298 ;
0058    299 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
0058    300 ;
0058    301 ; IMPLICIT INPUTS:
0058    302 ;
0058    303 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
0058    304 ;
0058    305 ; OUTPUT PARAMETERS:
0058    306 ;
0058    307 ;       NONE
0058    308 ;
0058    309 ; IMPLICIT OUTPUTS:
0058    310 ;
0058    311 ;       NONE
0058    312 ;
0058    313 ; COMPLETION CODES:
0058    314 ;
0058    315 ;       NONE
0058    316 ;
0058    317 ; SIDE EFFECTS:
0058    318 ;
0058    319 ;       NONE
0058    320 ;
0058    321 ;--
```

```
                                0058   323 VRCLREF::                                ;
                          08    BB  0058   324          PUSHR   #^M<R3>              ; SAVE SOME WORKING REGISTERS
          1C A2   FF000001    8F  CA 005A   325          BICL    #^XFF000001, -       ; CHECK FOR ILLEGAL FLAGS
                                0062   326                    CLI$L_DATA(R2)
                                0062   327
                       53       17  D0 0062   328 10$:     MOVL    #23,R3               ; UPPER FLAG NUMBER LIMIT
                                0065   329
             09 1C A2   53       E1  0065   330 20$:     BBC     R3, CLI$L_DATA(R2), 30$ ; EFN SPECIFIED ?
                                006A   331          $CLREF_S  R3                 ; YES, THEN GO CLEAR IT
                                0073   332
                       EF 53    F5  0073   333 30$:     SOBGTR  R3,20$               ; NEXT TILL DONE (DON'T DO 0)
                          08    BA  0076   334          POPR    #^M<R3>              ; RESTORE REGISTERS
                          05    05  0078   335          RSB                          ; RETURN
```

```
0079    337          .SBTTL  VRShowFlg Routine - Show control flags
0079    338 ;++
0079    339 ; FUNCTIONAL DESCRIPTION:
0079    340 ;
0079    341 ;       DISPLAYS OPERATOR CONTROL FLAGS WHICH ARE SET FOLLOWED BY
0079    342 ;       THOSE WHICH ARE CLEAR (I.E., DISPLAYS ALL CONTROL FLAGS)
0079    343 ;
0079    344 ; CALLING SEQUENCE:
0079    345 ;
0079    346 ;       BSBW    VRSHOWFLG
0079    347 ;
0079    348 ; INPUT PARAMETERS:
0079    349 ;
0079    350 ;       NONE
0079    351 ;
0079    352 ; IMPLICIT INPUTS:
0079    353 ;
0079    354 ;       DSA$GL_FLAGS = FLAGS WHICH ARE TO BE DISPLAYED
0079    355 ;
0079    356 ; OUTPUT PARAMETERS:
0079    357 ;
0079    358 ;       NONE
0079    359 ;
0079    360 ; IMPLICIT OUTPUTS:
0079    361 ;
0079    362 ;       NONE
0079    363 ;
0079    364 ; COMPLETION CODES:
0079    365 ;
0079    366 ;       NONE
0079    367 ;
0079    368 ; SIDE EFFECTS:
0079    369 ;
0079    370 ;       NONE
0079    371 ;
0079    372 ;--
```

```
                        0079   374 VRSHOWFLG::                                 ;
                  04 BB 0079   375         PUSHR   #^M<R2>                    ; SAVE A WORKING REGISTER
                        007B   376         $DS_CVTREG_S                    -  ; BUILD MNEMONIC STRING
                        007B   377             MSB=#LEFTMOSTFLAG,          -  ; FIRST BIT TO BE INTERPRETED
                        007B   378             DATA=L^DSA$GL_FLAGS,        -  ; Register location
                        007B   379             MNEADR=L^AFLAGMNE,          -  ; CONTROL STRING ADDRESS                [08]
                        007B   380             STRBUF=L^DS$GT_STRBUF,      -  ; MNEMONIC STRING BUFFER                [08]
                        007B   381             MAXLEN=#DS$K_STRBUF            ; SIZE OF ABOVE BUFFER
                        00A8   382
                        00A8   383         $Print -                           ; Display flags which are set          [11]
                        00A8   384                 #DS$K_Type_Command_Out, -; ... typecode number                 [11]
                        00A8   385                 #DS$K_PrintF, -            ; ... type of print                 [11]
                        00A8   386                 L^FMTSHOWFLGS, -           ; ... the format string             [11]
                        00A8   387                 #DS$GT_STRBUF              ; .. the converted register output  [11]
                        00C1   388
          52 0000FE00'EF D2 00C1   389         MCOML   L^DSA$GL_FLAGS, R2     ; Get cleared flags
                        00C8   390
                        00C8   391         $DS_CVTREG_S                    -  ; BUILD MNEMONIC STRING
                        00C8   392             MSB=#LEFTMOSTFLAG,          -  ; FIRST BIT TO BE INTERPRETED
                        00C8   393             DATA=R2,                    -  ; REGISTER LOCATION
                        00C8   394             MNEADR=L^AFLAGMNE,          -  ; CONTROL STRING ADDRESS                [08]
                        00C8   395             STRBUF=L^DS$GT_STRBUF,      -  ; MNEMONIC STRING BUFFER                [08]
                        00C8   396             MAXLEN=#DS$K_STRBUF            ; SIZE OF ABOVE BUFFER
                        00F1   397
                        00F1   398         $Print -                           ; Display flags which are clear        [11]
                        00F1   399                 #DS$K_Type_Command_Out, -; ... typecode number                 [11]
                        00F1   400                 #DS$K_PrintF, -            ; ... type of print                 [11]
                        00F1   401                 L^FMTSHOWFLGC, -           ; ... the format string             [11]
                        00F1   402                 #DS$GT_STRBUF              ; .. the converted register output  [11]
                        010A   403
                  04 BA 010A   404         POPR    #^M<R2>                    ; RESTORE REGISTERS
                     05 010C   405         RSB                                ; RETURN
```

```
                     407          .SBTTL  VRShowEF Routine - Show event flags
                     408 ;++
                     409 ; FUNCTIONAL DESCRIPTION:
                     410 ;
                     411 ;       DISPLAYS ALL OPERATOR ACCESSABLE EVENT FLAGS WHICH ARE
                     412 ;       CURRENTLY SET
                     413 ;
                     414 ; CALLING SEQUENCE:
                     415 ;
                     416 ;       BSBW    VRSHOWEF
                     417 ;
                     418 ; INPUT PARAMETERS:
                     419 ;
                     420 ;       NONE
                     421 ;
                     422 ; IMPLICIT INPUTS:
                     423 ;
                     424 ;       NONE
                     425 ;
                     426 ; OUTPUT PARAMETERS:
                     427 ;
                     428 ;       NONE
                     429 ;
                     430 ; IMPLICIT OUTPUTS:
                     431 ;
                     432 ;       DS$GL_EFCO = CLUSTER 0 EVENT FLAGS
                     433 ;
                     434 ; COMPLETION CODES:
                     435 ;
                     436 ;       NONE
                     437 ;
                     438 ; SIDE EFFECTS:
                     439 ;
                     440 ;       NONE
                     441 ;
                     442 ;--
```

```
                          010D   444 VRSHOWEF::                                  ;
                    04    BB    010D   445         PUSHR   #^M<R2>                ; SAVE A WORKING REGISTER
                          010F   446         $READEF_S  #0,L^DS$GL_EFC0     ; READ EVENT FLAG CLUSTER 0                [08]
     52   00000000'EF    FF 8F 9C 011E   447         ROTL    #-1,L^DS$GL_EFC0,R2    ; POSITION FOR CVTREG              [12]
                          0127   448
                          0127   449         $DS_CVTREG_S                   -       ; BUILD MNEMONIC STRING
                          0127   450             MSB=#22,                   -       ; FIRST BIT TO BE INTERPRETED
                          0127   451             DATA=R2,                   -       ; REGISTER LOCATION
                          0127   452             MNEADR=L^AEFNMNE,          -       ; CONTROL STRING ADDRESS           [08]
                          0127   453             STRBUF=L^DS$GT_STRBUF,     -       ; MNEMONIC STRING BUFFER           [08]
                          0127   454             MAXLEN=#DS$K_STRBUF                ; SIZE OF ABOVE BUFFER
                          0150   455
                          0150   456         $Print -                               ; Display flags which are set      [11]
                          0150   457                 #DS$K_Type_Command_Out, -      ; ... typecode number              [11]
                          0150   458                 #DS$K_PrintF, -                ; ... type of print                [11]
                          0150   459                 L^FMTSHOWEF, -                 ; ... the format string            [11]
                          0150   460                 #DS$GT_STRBUF                  ; .. the converted register output [11]
                          0169   461
                    04    BA    0169   462         POPR    #^M<R2>                ; RESTORE REGISTERS
                    05          016B   463         RSB                            ; RETURN
                          016C   464         .END
```

```
$$N                     =  00000002     D
$ENV                    =  00000002     D
AEFNMNE                    0000009E  R  D  02
AFLAGMNE                   00000051  R  D  02
BIT...                  =  00000004     D
CLI$K_BUFSIZ            =  00000100     D
CLI$K_SIZE                 00000444     D
CLI$L_ADDRESS              00000018     D
CLI$L_COMMAND              00000004     D
CLI$L_DATA                 0000001C     D
CLI$L_FLAGS                00000000     D
CLI$L_LAST                 00000024     D
CLI$L_NEXT                 00000030     D
CLI$L_PASS                 0000002C     D
CLI$L_SUBT                 00000028     D
CLI$L_TEST                 00000020     D
CLI$Q_BUFQWD               00000034     D
CLI$Q_FILE                 00000008     D
CLI$Q_SECTION              00000010     D
CLI$Q_TIME                 0000043C     D
CLI$T_BUFFER               0000003C     D
CLI$V_ADAPTER           =  00000018     D
CLI$V_ADR               =  0000000B     D
CLI$V_ASCII             =  00000013     D
CLI$V_BREAK             =  0000000A     D
CLI$V_BRIEF             =  0000001B     D
CLI$V_BYTE              =  0000000D     D
CLI$V_CLEAR             =  00000002     D
CLI$V_DEC               =  00000010     D
CLI$V_DEFAULT           =  0000000C     D
CLI$V_DEPOSIT           =  00000019     D
CLI$V_EVENT             =  00000008     D
CLI$V_EXAM              =  00000005     D
CLI$V_FLAGS             =  00000009     D
CLI$V_HEX               =  00000012     D
CLI$V_KERNEL            =  00000017     D
CLI$V_LOAD              =  00000006     D
CLI$V_LONG              =  0000000F     D
CLI$V_NOTNUF            =  00000001     D
CLI$V_OCT               =  00000011     D
CLI$V_PREG              =  0000001A     D
CLI$V_QA                =  00000007     D
CLI$V_QACKLOOPLOOPS     =  0000001C     D
CLI$V_QAERRORPRINTS     =  0000001B     D
CLI$V_QAMULTIPLEPASS    =  0000001F     D
CLI$V_QASUBTESTLOOPS    =  0000001E     D
CLI$V_QATESTLOOPS       =  0000001D     D
CLI$V_REG               =  00000014     D
CLI$V_REQUIRED          =  00000000     D
CLI$V_RUN               =  00000015     D
CLI$V_SET               =  00000003     D
CLI$V_SHOW              =  00000004     D
CLI$V_VALSEC            =  00000016     D
CLI$V_WORD              =  0000000E     D
DS$CVTREG                  ********  X  03
DS$GL_EFCO                 ********  X  03
DS$GT_STRBUF               ********  X  03
DS$K_PRINTB             =  00000002     D
DS$K_PRINTF             =  00000001     D
DS$K_PRINTI             =  00000000     D
DS$K_PRINTX             =  00000003     D
DS$K_STRBUF                ********  X  03
DS$K_TYPE_ABORT_PROGRAM =  00000014     D
DS$K_TYPE_ABORT_TEST    =  00000013     D
DS$K_TYPE_COMMAND_ERR   =  00000015     D
DS$K_TYPE_COMMAND_OUT   =  00000016     D
DS$K_TYPE_CRD_AUTOTEST  =  0000001A     D
DS$K_TYPE_DS_PROMPT     =  00000001     D
DS$K_TYPE_DS_START      =  0000001D     D
DS$K_TYPE_ERRDEV        =  00000008     D
DS$K_TYPE_ERRHARD       =  00000006     D
DS$K_TYPE_ERROR_BODY    =  00000009     D
DS$K_TYPE_ERROR_END     =  0000000A     D
DS$K_TYPE_ERRPREP       =  0000001B     D
DS$K_TYPE_ERRSOFT       =  00000007     D
DS$K_TYPE_ERRSUP        =  00000004     D
DS$K_TYPE_ERRSYS        =  00000005     D
DS$K_TYPE_ERR_HALT      =  0000000D     D
DS$K_TYPE_EXCEPTION     =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD=  0000000B     D
DS$K_TYPE_FIRST_PASS    =  00000011     D
DS$K_TYPE_GENERAL       =  00000000     D
DS$K_TYPE_GENERAL_ERROR =  00000003     D
DS$K_TYPE_NO_TESTS      =  00000012     D
DS$K_TYPE_PARAM_ERROR   =  0000001C     D
DS$K_TYPE_PROGRAM_END   =  00000010     D
DS$K_TYPE_PROGRAM_INFO  =  00000017     D
DS$K_TYPE_PROGRAM_START =  0000000F     D
DS$K_TYPE_QIO_INVADP    =  00000024     D
DS$K_TYPE_QIO_NODRIVER  =  00000022     D
DS$K_TYPE_QIO_WRONGVER  =  00000023     D
DS$K_TYPE_SCRIPT_ECHO   =  00000021     D
DS$K_TYPE_SCRIPT_PNF    =  0000001E     D
DS$K_TYPE_SCRIPT_PROMPT =  00000020     D
DS$K_TYPE_SCRIPT_SKIP   =  0000001F     D
DS$K_TYPE_SEQUENCE_ERROR=  00000019     D
DS$K_TYPE_START_ERR     =  00000018     D
DS$K_TYPE_START_LIST    =  00000025     D
DS$K_TYPE_SUMMARY       =  0000000E     D
DS$K_TYPE_USER_PROMPT   =  00000002     D
DSA$AL_APTMAIL             0000FE00     D
DSA$AT_APTTXT              0000FA00     D
DSA$GL_APTCOM              0000FE04     D
DSA$GL_DEVLEN              0000FE58     D
DSA$GL_ERRNO               0000FE44     D
DSA$GL_EVENT               0000FE48     D
DSA$GL_FLAGS               0000FE00     D
DSA$GL_MSGTYP              0000FE40     D
DSA$GL_PASSES              0000FE08     D
DSA$GL_PASSNO              0000FE54     D
DSA$GL_SECTNO              0000FE10     D
DSA$GL_SID                 0000FE14     D
DSA$GL_SUBTNO              0000FE4C     D
DSA$GL_TESTNO              0000FE50     D
```

```
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$M_ALLFLAGS                =   000037FD     D
DSA$M_BELL                    =   00000008     D
DSA$M_DFLTFLGS                    ********   X    03
DSA$M_HALT                    =   00000001     D
DSA$M_IE1                     =   00000010     D
DSA$M_IE2                     =   00000020     D
DSA$M_IE3                     =   00000040     D
DSA$M_IES                     =   00000080     D
DSA$M_LOOP                    =   00000004     D
DSA$M_OPER                    =   00001000     D
DSA$M_PROMPT                  =   00002000     D
DSA$M_QUICK                   =   00000100     D
DSA$M_TRACE                   =   00000400     D
DSA$M_VERIFY                  =   00000200     D
DSA$V_SEARCH                  =   0000000E     D
DSX$PRINT                         ********   X    03
FMTSHOWEF                         00000038 R   D    02
FMTSHOWFLGC                       0000001B R   D    02
FMTSHOWFLGS                       00000000 R   D    02
LEFTMOSTFLAG                  =   0000000E     D
SIZ...                        =   00000001     D
SYS$CLREF                         ********  GX    03
SYS$READEF                        ********  GX    03
SYS$SETEF                         ********  GX    03
VRCLREF                           00000058 RG  D    03
VRCLRFLG                          00000023 RG  D    03
VRSETEF                           00000037 RG  D    03
VRSETFLG                          00000000 RG  D    03
VRSETFLG_X                        00000022 R   D    03
VRSHOWEF                          0000010D RG  D    03
VRSHOWFLG                         00000079 RG  D    03
V_ENV                         =   00000001     D
V_LVL                         =   00000000     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000000DA (218.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	0000016C (364.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000002	460 (1)	387 (1) 402 (1) 460 (1)
$ENV	=00000002	105 (1)	
AEFNMNE	0000009E-R	133 (1)	454 (1)
AFLAGMNE	00000051-R	130 (1)	381 (1) 396 (1)
BIT...	=00000004	103 (1)	103 (1)
CLI$K_BUFSIZ	=00000100	101 (1)	101 (1)
CLI$K_SIZE	00000444	101 (1)	
CLI$L_ADDRESS	00000018	101 (1)	
CLI$L_COMMAND	00000004	101 (1)	
CLI$L_DATA	0000001C	101 (1)	#-180 (1) #-183 (1) #-185 (1) #-228 (1) #-231 (1) #-233 (1) #-276 (1) 280 (1) #-326 (1) 330 (1) 175 (1)
CLI$L_FLAGS	00000000	101 (1)	
CLI$L_LAST	00000024	101 (1)	
CLI$L_NEXT	00000030	101 (1)	
CLI$L_PASS	0000002C	101 (1)	
CLI$L_SUBT	00000028	101 (1)	
CLI$L_TEST	00000020	101 (1)	
CLI$Q_BUFQWD	00000034	101 (1)	
CLI$Q_FILE	00000008	101 (1)	
CLI$Q_SECTION	00000010	101 (1)	
CLI$Q_TIME	0000043C	101 (1)	
CLI$T_BUFFER	0000003C	101 (1)	
CLI$V_ADAPTER	=00000018	101 (1)	
CLI$V_ADR	=00000008	101 (1)	
CLI$V_ASCII	=00000013	101 (1)	
CLI$V_BREAK	=0000000A	101 (1)	
CLI$V_BRIEF	=0000001B	101 (1)	
CLI$V_BYTE	=0000000D	101 (1)	
CLI$V_CLEAR	=00000002	101 (1)	
CLI$V_DEC	=00000010	101 (1)	
CLI$V_DEFAULT	=0000000C	101 (1)	#-174 (1)
CLI$V_DEPOSIT	=00000019	101 (1)	
CLI$V_EVENT	=00000008	101 (1)	
CLI$V_EXAM	=00000005	101 (1)	
CLI$V_FLAGS	=00000009	101 (1)	
CLI$V_HEX	=00000012	101 (1)	
CLI$V_KERNEL	=00000017	101 (1)	
CLI$V_LOAD	=00000006	101 (1)	
CLI$V_LONG	=0000000F	101 (1)	
CLI$V_NOTNUF	=00000001	101 (1)	
CLI$V_OCT	=00000011	101 (1)	
CLI$V_PREG	=0000001A	101 (1)	
CLI$V_QA	=00000007	101 (1)	
CLI$V_QACKLOOPL OOPS	=0000001C	101 (1)	
CLI$V_QAERRORPR 'NTS	=0000001B	101 (1)	
CLI$V_QAMULTIPLL PASS	=0000001F	101 (1)	
CLI$V_QASUBTESTLOOPS	=0000001E	101 (1)	
CLI$V_QATESTLOOPS	=0000001D	101 (1)	
CLI$V_REG	=00000014	101 (1)	

```
CLI$V_REQUIRED                  =00000000      101     (1)
CLI$V_RUN                       =00000015      101     (1)
CLI$V_SET                       =00000003      101     (1)
CLI$V_SHOW                      =00000004      101     (1)
CLI$V_VALSEC                    =00000016      101     (1)
CLI$V_WORD                      =0000000E      101     (1)
DS$CVTREG                        00000000-XR                    381     (1)      396     (1)      454     (1)
DS$GL_EFCO                       00000000-XR                    446     (1)  #-447     (1)
DS$GT_STRBUF                     00000000-XR                    381     (1)  #-387     (1)      396     (1)  #-402     (1)
                                                                454     (1)  #-460     (1)
DS$K_PRINTB                     =00000002      103     (1)
DS$K_PRINTF                     =00000001      103     (1)  #-387     (1)  #-402     (1)  #-460     (1)
DS$K_PRINTI                     =00000000      103     (1)
DS$K_PRINTX                     =00000003      103     (1)
DS$K_STRBUF                      00000C00-XR                #-381     (1)  #-396     (1)  #-454     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014      103     (1)
DS$K_TYPE_ABORT_TEST            =00000013      103     (1)
DS$K_TYPE_COMMAND_ERR           =00000015      103     (1)
DS$K_TYPE_COMMAND_OUT           =00000016      103     (1)  #-387     (1)  #-402     (1)  #-460     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A      103     (1)
DS$K_TYPE_DS_PROMPT             =00000001      103     (1)
DS$K_TYPE_DS_START              =0000001D      103     (1)
DS$K_TYPE_ERRDEV                =00000008      103     (1)
DS$K_TYPE_ERRHARD               =00000006      103     (1)
DS$K_TYPE_ERROR_BODY            =00000009      103     (1)
DS$K_TYPE_ERROR_END             =0000000A      103     (1)
DS$K_TYPE_ERRPREP               =0000001B      103     (1)
DS$K_TYPE_ERRSOFT               =00000007      103     (1)
DS$K_TYPE_ERRSUP                =00000004      103     (1)
DS$K_TYPE_ERRSYS                =00000005      103     (1)
DS$K_TYPE_ERR_HALT              =0000000D      103     (1)
DS$K_TYPE_EXCEPTION             =0000000C      103     (1)
DS$K_TYPE_EXCEPTION_HEAD        =00000008      103     (1)
DS$K_TYPE_FIRST_PASS            =00000011      103     (1)
DS$K_TYPE_GENERAL               =00000000      103     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      103     (1)
DS$K_TYPE_NO_TESTS              =00000012      103     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      103     (1)
DS$K_TYPE_PROGRAM_END           =00000010      103     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      103     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      103     (1)
DS$K_TYPE_QIO_INVADP            =00000024      103     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      103     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      103     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      103     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      103     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      103     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      103     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      103     (1)
DS$K_TYPE_START_ERR             =00000018      103     (1)
DS$K_TYPE_START_LIST            =00000025      103     (1)
DS$K_TYPE_SUMMARY               =0000000E      103     (1)
DS$K_TYPE_USER_PROMPT           =00000002      103     (1)
DSA$GL_FLAGS                     0000FE00                   #-177     (1)  #-186     (1)  #-234     (1)  #-381     (1)
                                                            #-389     (1)
DSA$M_ALLFLAGS                  =000037FD      112     (1)  #-182     (1)  #-230     (1)
DSA$M_BELL                      =00000008                     109     (1)
```

```
DSA$M_DFLTFLGS          00000000-XR                     #-176   (1)
DSA$M_HALT             =00000001                        109     (1)
DSA$M_IE1              =00000010                        110     (1)
DSA$M_IE2              =00000020                        110     (1)
DSA$M_IE3              =00000040                        110     (1)
DSA$M_IES              =00000080                        112     (1)
DSA$M_LOOP             =00000004                        109     (1)
DSA$M_OPER             =00001000                        112     (1)
DSA$M_PROMPT           =00002000                        112     (1)
DSA$M_QUICK            =00000100                        111     (1)
DSA$M_TRACE            =00000400                        111     (1)
DSA$M_VERIFY           =00000200                        111     (1)
DSA$V_SEARCH           =0000000E                        107     (1)
DSX$PRINT               00000000-XR                     387     (1)     402     (1)     460     (1)
FMTSHOWEF               00000038-R      127     (1)     460     (1)
FMTSHOWFLGC             0000001B-R      124     (1)     402     (1)
FMTSHOWFLGS             00000000-R      121     (1)     387     (1)
LEFTMOSTFLAG           =0000000E        107     (1)     #-381   (1)     #-396   (1)
SYS$CLREF               00000000-XR                     331     (1)
SYS$READEF              00000000-XR                     446     (1)
SYS$SETEF               00000000-XR                     281     (1)
VRCLREF                 00000058-R      323     (1)
VRCLRFLG                00000023-R      227     (1)
VRSETEF                 00000037-R      273     (1)
VRSETFLG                00000000-R      173     (1)
VRSETFLG_X              00000022-R      188     (1)     #-178   (1)
VRSHOWEF                0000010D-R      444     (1)
VRSHOWFLG               00000079-R      374     (1)
V_ENV                  =00000001        102     (1)     105     (1)
V_LVL                  =00000000        102     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CLREF_S	1	331 (1)	331 (1)
$D1_PRINT_S	1	387 (1)	387 (1) 402 (1) 460 (1)
$DEF	1	103 (1)	
$DEFINI	1	100 (1)	100 (1)
$DS_CVTREG_S	1	377 (1)	377 (1) 392 (1) 450 (1)
$DS_DSADEF	5	100 (1)	100 (1)
$DS_TYPEDEF	4	103 (1)	103 (1)
$EQU	1	103 (1)	103 (1)
$EQULS1	1	103 (1)	103 (1)
$EQULST	1	103 (1)	103 (1)
$GBLINI	2		103 (1)
$PRINT	2	384 (1)	384 (1) 399 (1) 457 (1)
$PUSHADR	1	381 (1)	381 (1) 396 (1) 446 (1) 454 (1)
$READEF_S	1	446 (1)	446 (1)
$SETEF_S	1	281 (1)	281 (1)
$VIELDT	1	103 (1)	
CLIDEF	3	101 (1)	101 (1)
ENVDEF	1	102 (1)	102 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.09	00:00:00.24
Command processing	139	00:00:00.78	00:00:01.56
Pass 1	521	00:00:05.77	00:00:08.69
Symbol table sort	0	00:00:00.22	00:00:00.39
Pass 2	134	00:00:01.35	00:00:02.01
Symbol table output	18	00:00:00.13	00:00:00.54
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	35	00:00:00.53	00:00:00.78
Assembler run totals	890	00:00:08.91	00:00:14.25

The working set limit was 1000 pages.
29743 bytes (59 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 190 non-local and 9 local symbols.
464 source lines were read in Pass 1, producing 0 object records in Pass 2.
39 pages of virtual memory were used to define 21 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                       Macros defined
------------------                       --------------
DRB1:[DS.WORK]DIAG.MLB;955                          4
DRB1:[DS.WORK]DS.MLB;218                            3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1                       0
DMA1:[SYSO.SYSMAINT]DS.MLB;218                      0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953                    0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1                       0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2                   9
TOTALS (all libraries)                             16

305 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) FLAGS/UPDA=(FLAGS.UPD,FLAGS.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

Table of contents

(1) 38 CREATE I/O DRIVER FORK PROCESS
(2) 60 CREATE FORK PROCESS
(3) 88 SOFTWARE INTERRUPT FORK DISPATCHER

```
                        0000     .1          .TITLE  FRKCTL  *** FRKCTL QIO fork control
                        0000     .2          .IDENT  /06-03/
                        0000     .3          .LIST   MEB
                        0000     .4          .NLIST  CND
                        0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1976, 1977, 1978, 1979, 1980                               *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 9-AUG-76
                        0000     27 ;
                        0000     28 ; MODIFIED BY R.HEINEN ON 13-OCT-76; ALLOW FORK TO IPL 6
                        0000     29 ; FORK CREATION AND DISPATCHING
                        0000     30 ;
V54                     0000     .1 ; MODIFIED BY  T.L. SOUTTER  20-FEB-78
V54                     0000     .2 ;
V54                     0000     .3 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
V54                     0000     .4 ;
V54                     0000     .5 ;               Dave Butenhof   13-may-1980, Version 5.4
V54                     0000     .6 ;       02      Create .UPD file to modify VMS V2.0 source to
V54                     0000     .7 ;               Supervisor environment
03                      0000     .8 ;               Dave Butenhof, 23-feb-1981, version 6.3
03                      0000     .9 ;       03      Fix some truncation errors.
                        0000     31 ; MACRO LIBRARY CALLS
                        0000     32 ;
                        0000     33
03                      0000     .1
03                      0000     .5
                        0000     34          $FKBDEF                         ;DEFINE FKB OFFSETS
                        0000     35          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     36          $UCBDEF                         ;DEFINE UCB OFFSETS
03                      0000     .4
03                      0000     .5
03                  00000000     .6          .PSECT  SEP, SHR, EXE, WRT, LONG
03                      0000     .7
```

```
                                0000     38          .SBTTL  CREATE I/O DRIVER FORK PROCESS
                                0000     39 ;+
                                0000     40 ; EXE$IOFORK - CREATE I/O DRIVER FORK PROCESS
                                0000     41 ;
                                0000     42 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO CREATE A FORK PROCESS.
                                0000     43 ;
                                0000     44 ; INPUTS:
                                0000     45 ;
                                0000     46 ;       00(SP) = RETURN ADDRESS OF CALLER.
                                0000     47 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                0000     48 ;
                                0000     49 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0000     50 ;
                                0000     51 ; OUTPUTS:
                                C000     52 ;
                                0000     53 ;       ***TBS***
                                0000     54 ;-
                                0000     55
-1                              0000     57 EXE$IOFORK::                             ;CREATE I/O DRIVER FORK PROCESS
                00 58 A5   00   E5  0000     58          BBCC    #UCB$V_TIM,UCB$W_STS(R5),EXE$FORK ;DISABLE TIMEOUT
```

```
                                0005     60          .SBTTL  CREATE FORK PROCESS
                                0005     61 ;+
                                0005     62 ; EXE$FORK - CREATE FORK PROCESS
                                0005     63 ;
                                0005     64 ; THIS ROUTINE IS CALLED TO CREATE A FORK PROCESS.
                                0005     65 ;
                                0005     66 ; INPUTS:
                                0005     67 ;
                                0005     68 ;       00(SP) = RETURN ADDRESS OF CALLER.
                                0005     69 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                0005     70 ;
                                0005     71 ;       R5 = ADDRESS OF FORK BLOCK.
                                0005     72 ;
                                0005     73 ; OUTPUTS:
                                0005     74 ;
                                0005     75 ;       ***TBS***
                                0005     76 ;-
                                0005     77
                                0005     78 EXE$FORK::                                   ;CREATE FORK PROCESS
                   10 A5   53    7D     0005     79          MOVQ    R3,FKB$L_FR3(R5)    ;SAVE REGISTERS R3 AND R4
                      0C A5 8ED0     0009     80          POPL    FKB$L_FPC(R5)       ;SET FORK PROCESS PC
                   54    0B A5    9A     000D     81          MOVZBL  FKB$B_FIPL(R5),R4   ;GET FORK IPL
03              FFFFFFD0'EF44    7E     0011     .1          MOVAQ   L^SWI$GL_FQFL-<6*8>[R4], R3     ; Get address of fork queue
                         53             0018
-1                 04 B3   65    0E     0019     83          INSQUE  (R5),@4(R3)         ;INSERT FORK BLOCK IN FORK QUEUE
                           03   12     001D     84          BNEQ    10$                 ;IF NEQ NOT FIRST ENTRY IN QUEUE
                                001F     85          SOFTINT R4                  ;INITIATE SOFTWARE INTERRUPT
                      14   54    DA     001F                 MTPR    R4,S^#PR$_SIRR
                           05   0022     86 10$:     RSB                         ;
```

```
                                0023     88          .SBTTL  SOFTWARE INTERRUPT FORK DISPATCHER
                                0023     89 ;+
                                0023     90 ; EXE$FORKDSPTH - SOFTWARE INTERRUPT FORK DISPATCHER
                                0023     91 ;
                                0023     92 ; THIS ROUTINE IS AUTOMATICALLY VECTORED TO WHEN THE SOFTWARE INTERRUPT
                                0023     93 ; PRIORITY ARBITRATION LOGIC IN THE CENTRAL PROCESSOR DETECTS A PENDING
                                0023     94 ; INTERRUPT AT LEVEL 6, 7, 8, 9, 10, OR 11 AND THE CURRENT PRIORITY LEVEL IS
                                0023     95 ; LOWER THAN THE PENDING LEVEL. THE STATE OF THE STACK ON ENTRY IS:
                                0023     96 ;
                                0023     97 ;       00(SP) = INTERRUPT PC.
                                0023     98 ;       04(SP) = INTERRUPT PSL.
                                0023     99 ;
                                0023    100 ; FOR EACH OF THE LEVELS 6, 7, 8, 9, 10, AND 11, THERE EXISTS A QUEUE OF FORK
                                0023    101 ; BLOCKS WAITING TO BE PROCESSED. WHEN A FORK BLOCK IS ENTERED IN ITS
                                0023    102 ; CORRESPONDING QUEUE AND IT IS THE FIRST TO BE ENTERED (I. E. THE QUEUE
                                0023    103 ; WAS PREVIOUSLY EMPTY), A SOFTWARE INTERRUPT IS REQUESTED FOR THAT LEVEL.
                                0023    104 ; THE FORK DISPATCHER GAINS CONTROL AND EMPTIES THE QUEUE ONE ENTRY AT A
                                0023    105 ; TIME. AS EACH FORK IS DISPATCHED, REGISTERS R3 AND R4 ARE RESTORED FROM
                                0023    106 ; THE FORK BLOCK AND THE FORK PROCESS IS CALLED VIA A JSB INSTRUCTION.
                                0023    107 ; ON RETURN, THE FORK DISPATCHER RETRIEVES THE NEXT ENTRY FROM THE QUEUE
                                0023    108 ; AND REPEATS THE DISPATCHING OPERATION. THIS PROCESS CONTINUES UNTIL
                                0023    109 ; THERE ARE NO MORE FORKS TO DISPATCH AT WHICH TIME THE INTERRUPT IS
                                0023    110 ; DISMISSED.
                                0023    111 ;-
                                0023    112
                                0023    113          .ALIGN  LONG
                                0024    114 EXE$FORKDSPTH::                                 ;SOFTWARE INTERRUPT FORK DISPATCHER
                      3F    BB  0024    115          PUSHR   #^M<R0,R1,R2,R3,R4,R5>         ;SAVE FORK PROCESS REGISTER SET
                   50   12    DB  0026    116 10$:     MFPR    #PR$_IPL,R0                    ;READ CURRENT IPL
03              FFFFFFD0'EF40    7E  0029     .1          MOVAQ   L^SWI$GL_FQFL-<6*8>[R0], R1     ; Get address of fork queue
                      51        0030
-1                55   91    0F  0031    118          REMQUE  @(R1)+,R5                      ;REMOVE NEXT ENTRY FROM FORK QUEUE
                      2D    1D  0034    119          BVS     20$                            ;IF VS NO ENTRY REMOVED
                53   10 A5    7D  0036    120          MOVQ    FKB$L_FR3(R5),R3               ;RESTORE REGISTERS R3 AND R4
                                003A    121
                                003A    122 ;
                                003A    123 ; DISPATCH FORK PROCESS WITH:
                                003A    124 ;
                                003A    125 ;       R0 THRU R2 = SCRATCH REGISTERS.
                                003A    126 ;       R3 AND R4 = RESTORED FROM FORK BLOCK.
                                003A    127 ;       R5 = ADDRESS OF FORK BLOCK.
                                003A    128 ;
                                003A    129
             51    0C A5    D0  003A    130          MOVL    FKB$L_FPC(R5),R1               ;SAVE DISPATCH ADDRESS
                      03    BB  003F    131          PUSHR   #^M<R0,R1>                     ;SAVE IPL AND DISPATCH ADDRESS
                      61    16  0040    132          JSB     (R1)                           ;DISPATCH FORK
                      03    BA  0042    133          POPR    #^M<R0,R1>                     ;RESTORE IPL, DISPATCH ADDRESS
                   52   12    DB  0044    134          MFPR    #PR$_IPL,R2                    ;GET EXIT IPL
                   52   50    D1  0047    135          CMPL    R0,R2                          ;EXIT IPL MUST EQUAL ENTRY
                        DA    13  004A    136          BEQL    10$                            ;GO AGAIN
                                004C    137
                                004C    138          BUG_CHECK BADFORKIPL,FATAL             ;BAD FORK EXIT INTERRUPT PRIORITY LEVEL
                 00000000'9F    16  004C                JSB     @#BUG$CHECK
              4F 46 44 41 42 00'  0052                .ASCIC  "BADFORKIPL,FATAL"
              2C 4C 50 49 4B 52   0058
                 4C 41 54 41 46   005E
                                10  0052
```

```
                     3F   BA  0063   139 20$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ;RESTORE FORK PROCESS REGISTER SET
                          02  0065   140         REI                             ;
                              0066   141
                              0066   142         .END
```

BUG$CHECK	********	X	02	UCB$L_CPID	0000005C	D	UCB$W_DIRSEQ	00000088	D		
EXE$FORK	00000005	RG D	02	UCB$L_CRB	00000020	D	UCB$W_DSTADDR	00000018	D		
EXE$FORKDSPTH	00000024	RG D	02	UCB$L_DDB	00000024	D	UCB$W_DX_BCR	000000A4	D		
EXE$IOFORK	00000000	RG D	02	UCB$L_DEVCHAR	00000034	D	UCB$W_EC1	00000090	D		
FKB$B_FIPL	0000000B	D		UCB$L_DEVDEPEND	0000003C	D	UCB$W_EC2	00000092	D		
FKB$B_TYPE	0000000A	D		UCB$L_DPC	00000080	D	UCB$W_ERRCNT	00000072	D		
FKB$C_LENGTH	00000018	D		UCB$L_DUETIM	0000005C	D	UCB$W_FUNC	0000007E	D		
FKB$K_LENGTH	00000018	D		UCB$L_DX_BFPNT	0000009C	D	UCB$W_MB_SEED	00000000	D		
FKB$L_FPC	0000000C	D		UCB$L_DX_BUF	00000098	D	UCB$W_MSGCNT	00000016	D		
FKB$L_FQBL	00000004	D		UCB$L_DX_RXDB	000000A0	D	UCB$W_MSGMAX	00000014	D		
FKB$L_FQFL	00000000	D		UCB$L_EMB	00000078	D	UCB$W_NT_CHAN	0000007C	D		
FKB$L_FR3	00000010	D		UCB$L_FIRST	00000014	D	UCB$W_OFFSET	0000008A	D		
FKB$L_FR4	00000014	D		UCB$L_FPC	0000000C	D	UCB$W_REFC	00000050	D		
FKB$W_SIZE	00000008	D		UCB$L_FQBL	00000004	D	UCB$W_SIZE	00000008	D		
PR$_IPL =	00000012	D		UCB$L_FQFL	00000000	D	UCB$W_SRCADDR	0000001A	D		
PR$_SIRR =	00000014	D		UCB$L_FR3	00000010	D	UCB$W_STS	00000058	D		
SIZ... =	00000002	D		UCB$L_FR4	00000014	D	UCB$W_TT_DESIZE	00000045	D		
SWI$GL_FQFL	********	X	02	UCB$L_IOQBL	00000044	D	UCB$W_UNIT	00000048	D		
UCB$B_AMOD	00000053	D		UCB$L_IOQFL	00000040	D	UCB$W_VPROT	0000001A	D		
UCB$B_CEX	00000077	D		UCB$L_IRP	0000004C	D					
UCB$B_CM1	0000004A	D		UCB$L_LINK	0000002C	D					
UCB$B_CM2	0000004B	D		UCB$L_LOGADR	00000064	D					
UCB$B_DEVCLASS	00000038	D		UCB$L_MAXBLOCK	00000084	D					
UCB$B_DEVTYPE	00000039	D		UCB$L_MB_MBX	0000007C	D					
UCB$B_DIPL	00000052	D		UCB$L_MB_PORT	0000008C	D					
UCB$B_DX_SCTCNT	000000A6	D		UCB$L_MB_RAST	00000078	D					
UCB$B_ERTCNT	00000070	D		UCB$L_MB_SHB	00000080	D					
UCB$B_ERTMAX	00000071	D		UCB$L_MB_WAST	00000074	D					
UCB$B_FEX	00000076	D		UCB$L_MB_WIOQBL	00000088	D					
UCB$B_FIPL	0000000B	D		UCB$L_MB_WIOQFL	00000084	D					
UCB$B_LOCSRV	0000003C	D		UCB$L_MEDIA	0000008C	D					
UCB$B_OFFNDX	00000094	D		UCB$L_NT_DATSSB	00000074	D					
UCB$B_OFFRTC	00000095	D		UCB$L_NT_INTSSB	00000078	D					
UCB$B_REMSRV	0000003D	D		UCB$L_OPCNT	00000060	D					
UCB$B_SECTORS	0000003C	D		UCB$L_OWNUIC	0000001C	D					
UCB$B_SLAVE	00000074	D		UCB$L_PID	00000028	D					
UCB$B_SPR	00000075	D		UCB$L_RQBL	00000004	D					
UCB$B_STATE	00000052	D		UCB$L_RQFL	00000000	D					
UCB$B_TRACKS	0000003D	D		UCB$L_SVAPTE	00000068	D					
UCB$B_TT_CRFILL	0000009D	D		UCB$L_SVPN	00000064	D					
UCB$B_TT_DECRF	000000A1	D		UCB$L_TT_DECHAR	000000A8	D					
UCB$B_TT_DELFF	000000A2	D		UCB$L_TT_RDUE	0000008C	D					
UCB$B_TT_DESPEE	000000A0	D		UCB$L_TT_RTIMOU	00000088	D					
UCB$B_TT_DETYPE	000000A4	D		UCB$L_VCB	00000030	D					
UCB$B_TT_LFFILL	0000009E	D		UCB$L_PARTNER	0000000C	D					
UCB$B_TT_SPEED	0000009C	D		UCB$V_TIM =	00000000	D					
UCB$B_TYPE	0000000A	D		UCB$W_BCNT	0000006E	D					
UCB$B_VERTSZ	0000003F	D		UCB$W_BCR	00000096	D					
UCB$C_LENGTH	00000074	D		UCB$W_BOFF	0000006C	D					
UCB$C_MB_LENGTH	00000090	D		UCB$W_BUFQUO	00000018	D					
UCB$C_TT_LENGTH	000000BC	D		UCB$W_BYTESTOGO	0000003E	D					
UCB$K_LENGTH	00000074	D		UCB$W_CHARGE	0000004A	D					
UCB$K_MB_LENGTH	00000090	D		UCB$W_CYLINDERS	0000003E	D					
UCB$K_TT_LENGTH	000000BC	D		UCB$W_DA	0000008C	D					
UCB$L_AMB	00000054	D		UCB$W_DC	0000008E	D					
UCB$L_ASTQBL	00000010	D		UCB$W_DEVBUFSIZ	0000003A	D					
UCB$L_ASTQFL	0000000C	D		UCB$W_DEVSTS	0000005A	D					

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000066 (102.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BUG$CHECK	00000000-XR		138 (3)
EXE$FORK	00000005-R	78 (2)	#-58 (1)
EXE$FORKDSPTH	00000024-R	114 (3)	
EXE$IOFORK	00000000-R	57 (1)	
FKB$B_FIPL	0000000B		#-81 (2)
FKB$L_FPC	0000000C		#-130 (3) #-80 (2)
FKB$L_FR3	00000010		#-120 (3) #-79 (2)
PR$_IPL	=00000012		#-116 (3) #-134 (3)
PR$_SIRR	=00000014		#-85 (2)
SWI$GL_FQFL	00000000-XR		116.1 (3) 81.1 (2)
UCB$V_TIM	=00000000		#-58 (1)
UCB$W_STS	00000058		58 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...			
$DEFINI	1	34 (1)	34 (1)	35 (1)	36 (1)	
$FKBDEF	1	34 (1)	34 (1)			
$PRDEF	4	35 (1)	35 (1)			
$UCBDEF	10	36 (1)	36 (1)			
BUG_CHECK	1	138 (3)	138 (3)			
SOFTINT	1	85 (2)	85 (2)			

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.13	00:00:00.33
Command processing	139	00:00:00.76	00:00:01.84
Pass 1	520	00:00:05.54	00:00:09.45
Symbol table sort	0	00:00:00.37	00:00:00.47
Pass 2	66	00:00:01.14	00:00:02.93
Symbol table output	2	00:00:00.07	00:00:00.11
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	15	00:00:00.10	00:00:00.10
Assembler run totals	787	00:00:08.14	00:00:15.25

The working set limit was 1000 pages.
26247 bytes (52 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 277 non-local and 3 local symbols.
164 source lines were read in Pass 1, producing 0 object records in Pass 2.
27 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	2
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	10

478 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) FRKCTL/UPDA=(FRKCTL.UPD,FRKCTL.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA

Table of contents

(2) 41 GET TIME

```
                                        .TITLE  GETTIM - SYSTEM SERVICE GET TIME
                                        .IDENT  /01/

                                ;
                                ; COPYRIGHT (C) 1976, 1980
                                ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                                ;
                                ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
                                ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
                                ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
                                ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
                                ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
                                ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
                                ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
                                ;
                                ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
                                ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
                                ; EQUIPMENT CORPORATION.
                                ;
                                ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                                ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                ;
                                ; D. N. CUTLER 30-SEP-76
                                ;
                                ; SYSTEM SERVICE GET TIME
                                ;
                                ; MACRO LIBRARY CALLS
                                ;

                                        $SSDEF                          ;DEFINE SYSTEM STATUS VALUES

                                ;
                                ; LOCAL SYMBOLS
                                ;
                                ; ARGUMENT LIST OFFSET DEFINITIONS
                                ;

00000004                        TIMADR=4                                ;ADDRESS OF QUADWORD TO RECEIVE TIME
```

```
                                0000     41          .SBTTL  GET TIME
                                0000     42 ;+
                                0000     43 ; EXE$GETTIM - GET TIME
                                0000     44 ;
                                0000     45 ; THIS SERVICE PROVIDES THE CAPABILITY TO RETRIEVE THE CURRENT SYSTEM TIME
                                0000     46 ; IN 64 BIT FORMAT.
                                0000     47 ;
                                0000     48 ; INPUTS:
                                0000     49 ;
                                0000     50 ;       TIMADR(AP) = ADDRESS OF QUADWORD THAT IS TO RECEIVE TIME.
                                0000     51 ;
                                0000     52 ; OUTPUTS:
                                0000     53 ;
                                0000     54 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO RETRIEVE SYSTEM TIME.
                                C000     55 ;
                                0000     56 ;               R0 = SS$_ACCVIO - QUADWORD TO RECEIVE TIME CANNOT BE
                                0000     57 ;                    WRITTEN BY CALLING ACCESS MODE.
                                0000     58 ;
                                0000     59 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0000     60 ;
                                0000     61 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0000     62 ;-
                                0000     63
                           00000000     64          .PSECT  SEP, SHR, WRT, EXE, LONG
                                0000     65 EXE$GETTIM::                             ;GET TIME
                           0000  0000     66          .WORD   0                       ;ENTRY MASK
                 51    04 AC    D0  0002     67          MOVL    TIMADR(AP),R1       ;GET ADDRESS OF QUADWORD TO RECEIVE TIME
                    50    0C    3C  0006     68          MOVZWL  #SS$_ACCVIO,R0      ;ASSUME QUADWORD NOT WRITABLE
                                    0009     69          IFNOWRT #8,(R1),10$         ;CAN QUADWORD BE WRITTEN?
             61  00000000'EF    7D  000F     70          MOVQ    EXE$GQ_SYSTIME,(R1) ;STORE SYSTEM TIME IN QUADWORD
                    50    01    3C  0016     71          MOVZWL  #SS$_NORMAL,R0      ;SET NORMAL COMPLETION
                                04  0019     72 10$:     RET                         ;
                                    001A     73
                                    001A     74          .END
```

```
EXE$GETTIM                      00000000 RG D  02
EXE$GQ_SYSTIME                  ******** X    02
SS$_ACCVIO                    = 0000000C     D
SS$_NORMAL                    = 00000001     D
TIMADR                        = 00000004     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	0000001A (26.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
EXE$GETTIM	00000000-R	65 (2)	
EXE$GQ_SYSTIME	00000000-XR		#-70 (2)
SS$_ACCVIO	=0000000C		#-68 (2)
SS$_NORMAL	=00000001		#-71 (2)
TIMADR	=00000004	39 (2)	#-67 (2)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	31 (2)	31 (2)
$SSDEF	21	31 (2)	31 (2)
IFNOWRT	1	69 (2)	69 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.10	00:00:00.28
Command processing	151	00:00:00.75	00:00:01.75
Pass 1	230	00:00:03.73	00:00:05.36
Symbol table sort	0	00:00:00.58	00:00:00.70
Pass 2	46	00:00:00.58	00:00:00.73
Symbol table output	1	00:00:00.02	00:00:00.02
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	7	00:00:00.05	00:00:00.05
Assembler run totals	468	00:00:05.85	00:00:08.95

The working set limit was 1000 pages.
19064 bytes (38 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 405 non-local and 1 local symbols.
74 source lines were read in Pass 1, producing 0 object records in Pass 2.
10 pages of virtual memory were used to define 8 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	5

468 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) GETTIM/UPDA=(GETTIM.UPD,GETTIM.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 53 GET I/O CHANNEL DEVICE INFORMATION

B 1 check for legal event flag num
C 1 set event flag routine
D 1 set event flag routine
E 1 Clear event flag routine
F 1 Clear event flag routine
G 1 post event flag
H 1 post event flag
I 1 read event flags routine
J 1 read event flags routine
K 1 wait for single event flag rou
L 1 wait for single event flag rou
M 1 wait for logical or of event f
N 1 wait for logical or of event f
B 2 wait for logical and of event
C 2 wait for logical and of event
D 2 wait for logical and of event
E 2 *** EXCEPT Machine Exception h
F 2 *** EXCEPT Machine Exception h
G 2 *** EXCEPT Machine Exception h
H 2 *** EXCEPT Machine Exception h
I 2 Libraries, External & Equated
J 2 Work Psect Declarations
K 2 Data Psect Declarations
L 2 Data Psect Declarations
M 2 Data Psect Declarations
N 2 Data Psect Declarations
B 3 Data Psect Declarations
C 3 Data Psect Declarations
D 3 Data Psect Declarations
E 3 Data Psect Declarations
F 3 COND_HANDLR ROUTINE - HANDLE U
G 3 COND_HANDLR ROUTINE - HANDLE U
H 3 COND_HANDLR ROUTINE - HANDLE U
I 3 DSX$PrintSig Routine - Format
J 3 DSX$PrintSig Routine - Format
K 3 DSX$PrintSig Routine - Format
L 3 DSX$PrintSig Routine - Format
M 3 DSX$PrintSig Routine - Format
N 3 DSX$PrintSig Routine - Format
B 4 DSX$PrintSig Routine - Format
C 4 DSX$PrintSig Routine - Format
D 4 FIND_USERPC Find USER PC on st
E 4 FIND_USERPC Find USER PC on st
F 4 TST$MChk Routine - Machine Che
G 4 TST$MChk Routine - Machine Che
H 4 HARDWARE EXCEPTION ENTRY POINT
I 4 HARDWARE EXCEPTION ENTRY POINT
J 4 HARDWARE EXCEPTION ENTRY POINT
K 4 HARDWARE EXCEPTION ENTRY POINT
L 4 HARDWARE EXCEPTION ENTRY POINT
M 4 HARDWARE EXCEPTION ENTRY POINT
N 4 HARDWARE EXCEPTION ENTRY POINT
B 5 SEARCH FOR CONDITION HANDLER
C 5 SEARCH FOR CONDITION HANDLER
D 5 SEARCH FOR CONDITION HANDLER
E 5 DSX$SETPRIEXV - ESTABLISH PRIM
F 5 Symbol table
G 5 Symbol table
H 5 Symbol table
I 5 Symbol table
J 5 Symbol table
K 5 Psect synopsis
L 5 Cross reference
M 5 Cross reference
N 5 Cross reference
B 6 Cross reference
C 6 Cross reference
D 6 Cross reference
E 6 Cross reference
F 6 Cross reference
G 6 Cross reference
H 6 Cross reference
I 6 Cross reference
J 6 *** - FORMATTED ASCII OUTPUT
K 6 *** - FORMATTED ASCII OUTPUT
L 6 *** - FORMATTED ASCII OUTPUT
M 6 Libraries, Equated Symbols
N 6 Libraries, Equated Symbols
B 7 Libraries, Equated Symbols
C 7 FAO - Main program
D 7 FAO - Main program
E 7 FAO - Main program
F 7 FAO - Main program
G 7 FAO - Main program
H 7 GETCHAR - Routine to get next
I 7 GETCHAR - Routine to get next
J 7 GETCOUNT - Routine to get repe
K 7 GETCOUNT - Routine to get repe
L 7 CVTASC - Insert ASCII string
M 7 CVTASC - Insert ASCII string
N 7 CVTASC - Insert ASCII string
B 8 CVTNUM - Convert numeric param
C 8 CVTNUM - Convert numeric param
D 8 CVTNUM - Convert numeric param
E 8 CVTNUM - Convert numeric parm
F 8 QUICKSERVE - Small service rou
G 8 QUICKSERVE - Small service rou
H 8 QUICKSERVE - Small service rou
I 8 PERCENT - Time directives and
J 8 PERCENT - Time directives and
K 8 PERCENT - Time directives and
L 8 HANDLER - Condition handler
M 8 HANDLER - Condition handler
N 8 Symbol table
B 9 Psect synopsis
C 9 Cross reference
D 9 Cross reference
E 9 Cross reference
F 9 - FILES11 LEVEL 1 & 2 FILE REA
G 9 - FILES11 LEVEL 1 & 2 FILE REA
H 9 - FILES11 LEVEL 1 & 2 FILE REA
I 9 DECLARATIONS
J 9 DECLARATIONS
K 9 DECLARATIONS
L 9 FIL$OPENFILE - RETURN FILE HEA
M 9 FIL$OPENFILE - RETURN FILE HEA
N 9 FIL$OPENFILE - RETURN FILE HEA
B 10 FIL$OPENFILE - RETURN FILE HEA
C 10 FIL$OPENFILE - RETURN FILE HEA
D 10 FIL$CACHE_INIT - INIT FILEREAD
E 10 FIL$CACHE_INIT - INIT FILEREAD
F 10 FIL$CACHE_TRUNC - TRUNCATE FIL
G 10 ASSIGN_DEV - ASSIGN A DEVICE A
H 10 STORE3DIGITS - STORE 3 ASCII D
I 10 FORMDIRSTRING - GET A DIRECTOR
J 10 FORMDIRSTRING - GET A DIRECTOR
K 10 MOUNT - MOUNT THE VOLUME, INIT
L 10 MOUNT - MOUNT THE VOLUME, INIT
M 10 MOUNT - MOUNT THE VOLUME, INIT
N 10 FINDFILID - FIND FILE ID FOR S
B 11 FINDFILID - FIND FILE ID FOR S
C 11 FINDFILID - FIND FILE ID FOR S
D 11 FINDFILID - FIND FILE ID FOR S
E 11 FINDFILID - FIND FILE ID FOR S
F 11 FINDFILID - FIND FILE ID FOR S
G 11 FIL$FINDFILID - STRUCTURE LEVE
H 11 FIL$FINDFILID - STRUCTURE LEVE
I 11 FIL$FINDFILID - STRUCTURE LEVE
J 11 FIL$FINDFILID - STRUCTURE LEVE
K 11 FIL$FINDFILID - STRUCTURE LEVE
L 11 READ_DIR_LBN - READ NEXT DIREC
M 11 RDCHKFILHDR - READ AND CHECK F
N 11 RDCHKFILHDR - READ AND CHECK F
B 12 RDCHKFILHDR - READ AND CHECK F
C 12 READVBN, WRITEVBN - READ/WRITE
D 12 READVBN, WRITEVBN - READ/WRITE
E 12 INIRTRVPTRSCAN - INITIALIZE RE
F 12 GETRTRVPTR - CONVERT NEXT RETR
G 12 GETRTRVPTR - CONVERT NEXT RETR
H 12 STATBLK - GET FILE STATISTICS
I 12 STATBLK - GET FILE STATISTICS
J 12 STATBLK - GET FILE STATISTICS
K 12 FIL$CHKFILHDR - CHECK FILE HEA
L 12 FIL$CHKFILHDR - CHECK FILE HEA
M 12 CHECKSUM - VALIDATE A CHECKSUM
N 12 Symbol table
B 13 Symbol table
C 13 Psect synopsis
D 13 Cross reference
E 13 Cross reference
F 13 Cross reference
G 13 Cross reference
H 13 Cross reference
I 13 Cross reference
J 13 *** FLAGS Set/Clear/Show flags
K 13 *** FLAGS Set/Clear/Show flags
L 13 *** FLAGS Set/Clear/Show flags
M 13 *** FLAGS Set/Clear/Show flags
N 13 Libraries, Macros, Equated Sym
B 14 Data Psect Declarations
C 14 VRSetFlg Routine - Set control
D 14 VRSetFlg Routine - Set control
E 14 VRClrFlg Routine - Clear contr
F 14 VRClrFlg Routine - Clear contr
G 14 VRSetEF Routine - Set event fl
H 14 VRSetEF Routine - Set event fl
I 14 VRClrEF Routine - Clear event
J 14 VRClrEF Routine - Clear event
K 14 VRShowFlg Routine - Show contr
L 14 VRShowFlg Routine - Show contr

M 14 VRShowEF Routine - Show event
N 14 VRShowEF Routine - Show event
B 15 Symbol table
C 15 Symbol table
D 15 Cross reference
E 15 Cross reference
F 15 Cross reference
G 15 Cross reference
H 15 Cross reference
I 15 *** FRKCTL QIO fork control
J 15 *** FRKCTL QIO fork control
K 15 CREATE I/O DRIVER FORK PROCESS
L 15 CREATE FORK PROCESS
M 15 SOFTWARE INTERRUPT FORK DISPAT
N 15 SOFTWARE INTERRUPT FORK DISPAT
B 16 Symbol table
C 16 Psect synopsis
D 16 Cross reference
E 16 Cross reference
F 16 - SYSTEM SERVICE GET TIME
G 16 - SYSTEM SERVICE GET TIME
H 16 GET TIME
I 16 Symbol table
J 16 Cross reference
K 16 Cross reference
L 16 *** GTCHAN Get channel informa

```
                     0000     1          .TITLE  GTCHAN  *** GTCHAN Get channel information
                     0000     2          .IDENT  /02-01/
                     0000     3
                     0000     4 ;
                     0000     5 ; COPYRIGHT (C) 1977,, 1980
                     0000     6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                     0000     7 ;
                     0000     8 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
                     0000     9 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
                     0000    10 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
                     0000    11 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
                     0000    12 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
                     0000    13 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
                     0000    14 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
                     0000    15 ;
                     0000    16 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
                     0000    17 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
                     0000    18 ; EQUIPMENT CORPORATION.
                     0000    19 ;
                     0000    20 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                     0000    21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                     0000    22 ;
                     0000    23 ; D. N. CUTLER 22-SEP-76
                     0000    24 ;
                     0000    25 ; MODIFICATION HISTORY
                     0000    26 ;                N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
                     0000    27 ;       01       DIAGNOSTIC SUPERVISOR INTEGRATION
                     0000    28 ;
                     0000    29 ; SYSTEM SERVICE GET I/O CHANNEL DEVICE INFORMATION
                     0000    30
                     0000    31 ;
                     0000    32 ; MACRO LIBRARY CALLS
                     0000    33 ;
                     0000    34
                     0000    35          $CCBDEF                             ;DEFINE CCB OFFSETS
                     0000    36          $DDBDEF                             ;DEFINE DDB OFFSETS
                     0000    37          $DEVDEF                             ;DEFINE DEVICE CHARACTERISTICS
                     0000    38          $SSDEF                              ;DEFINE SYSTEM STATUS VALUES
                     0000    39          $UCBDEF                             ;DEFINE UCB OFFSETS
                     0000    40
                     0000    41 ;
                     0000    42 ; LOCAL SYMBOLS
                     0000    43 ;
                     0000    44 ; ARGUMENT LIST OFFSET DEFINITIONS
                     0000    45 ;
                     0000    46
00000004             0000    47 CHAN=4                                      ;I/O CHANNEL NUMBER
00000008             0000    48 PRILEN=8                                    ;ADDRESS TO STORE LENGTH OF PRIMARY STRING
0000000C             0000    49 PRIBUF=12                                   ;ADDRESS OF PRIMARY BUFFER DESCRIPTOR
00000010             0000    50 SCDLEN=16                                   ;ADDRESS TO STORE LENGTH OF SECONDARY STRING
00000014             0000    51 SCDBUF=20                                   ;ADDRESS OF SECONDARY BUFFER DESCRIPTOR
```

```
                                        53          .SBTTL  GET I/O CHANNEL DEVICE INFORMATION
                                        54 ;+
                                        55 ; EXE$GTCHAN - GET I/O CHANNEL DEVICE INFORMATION
                                        56 ;
                                        57 ; THIS SERVICE PROVIDES THE CAPABILITY TO RETRIEVE INFORMATION ABOUT A
                                        58 ; DEVICE THAT IS ASSIGNED TO A CHANNEL AND ITS ASSOCIATED DEVICE IF ANY.
                                        59 ;
                                        60 ; INPUTS:
                                        61 ;
                                        62 ;       CHAN(AP) = I/O CHANNEL NUMBER.
                                        63 ;       PRILEN(AP) = ADDRESS TO STORE LENGTH OF PRIMARY STRING.
                                        64 ;       PRIBUF(AP) = ADDRESS OF PRIMARY BUFFER DESCRIPTOR.
                                        65 ;       SCDLEN(AP) = ADDRESS TO STORE LENGTH OF SECONDARY STRING.
                                        66 ;       SCDBUF(AP) = ADDRESS OF SECONDARY BUFFER DESCRIPTOR.
                                        67 ;
                                        68 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                        69 ;
                                        70 ; OUTPUTS:
                                        71 ;
                                        72 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO RETRIEVE DEVICE INFORMATION.
                                        73 ;
                                        74 ;               R0 = SS$_ACCVIO - PRIMARY OR SECONDARY BUFFER DESCRIPTOR
                                        75 ;                       CANNOT BE READ BY CALLING ACCESS MODE, OR PRIMARY
                                        76 ;                       OR SECONDARY BUFFER CANNOT BE WRITTEN BY CALLING
                                        77 ;                       ACCESS MODE.
                                        78 ;
                                        79 ;               R0 = SS$_IVCHAN - INVALID CHANNEL NUMBER SPECIFIED.
                                        80 ;
                                        81 ;               R0 = SS$_NOPRIV - SPECIFIED CHANNEL IS NOT ASSIGNED TO A
                                        82 ;                       DEVICE OR THE CALLING ACCESS MODE DOES NOT HAVE
                                        83 ;                       PRIVILEGE TO ACCESS THE CHANNEL.
                                        84 ;
                                        85 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                        86 ;
                                        87 ;               R0 = SS$_BUFFEROVF - NORMAL COMPLETION, ALL CHARACTERISTIC
                                        88 ;                       INFORMATION DID NOT FIT IN SPECIFIED BUFFER(S).
                                        89 ;
                                        90 ;               R0 = SS$_NORMAL - NORMAL COMPLETION, ALL CHARACTERISTIC
                                        91 ;                       INFORMATION TRANSFERED.
                                        92 ;-
                                        93
                                 00000000 94          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                                        95
                                 01FC   96 .ENTRY   EXE$GTCHAN,^M<R2,R3,R4,R5,R6,R7,R8>
                                        97
                                        98          MOVZWL  CHAN(AP),R0             ;GET CHANNEL NUMBER
                                        99          BSBW    IOC$VER.FYCHAN          ;VERIFY CHANNEL NUMBER
                                       100          BLBC    R0,40$                  ;IF LBC INVALID CHANNEL
                                       101          MOVL    CCB$L_UCB(R1),R5        ;GET DEVICE UNIT UCB ADDRESS
                                       102          MOVL    R5,R4                   ;ASSUME ASSOCIATED UCB NOT SPOOL DEVICE
                                       103          BBC     #DEV$V_SPL,UCB$L_DEVCHAR(R5),10$ ;IF CLR, THEN DEVICE NOT SPOOLED
                                       104          MOVL    UCB$L_AMB(R5),R4        ;GET INTERMEDIATE DEVICE UCB ADDRESS
                                       105 10$:     MOVZWL  #SS$_NORMAL,R6          ;ASSUME ALL INFO CAN BE RETURNED
                                       106          MOVQ    PRILEN(AP),R7           ;GET PRIMARY BUFFER PARAMETERS
                                       107          BSBB    FILBUF                  ;FILL PRIMARY BUFFER
                                       108          MOVL    UCB$L_AMB(R5),R4        ;ANY ASSOCIATED UCB?
                                       109          BEQL    20$                     ;IF EQL NO
```

```
                03 34 A5    06   E1  002A   110          BBC     #DEV$V_SPL,UCB$L_DEVCHAR(R5),30$ ;IF CLR, THEN DEVICE NOT SPOOLED
                      54    55   D0  002F   111 20$:      MOVL    R5,R4                            ;SET SECONDARY DEVICE UCB ADDRESS
                   57   10 AC   7D  0032   112 30$:      MOVQ    SCDLEN(AP),R7                    ;GET SECONDARY BUFFER PARAMETERS
                              04   10  0036   113          BSBB    FILBUF                           ;FILL SECONDARY BUFFER
                         50   56   D0  0038   114          MOVL    R6,R0                            ;SET FINAL REQUEST STATUS
                              04   04  003B   115 40$:      RET                                      ;
                                        003C   116
                                        003C   117 ;
                                        003C   118 ; SUBROUTINE TO FILL CHARACTERISTIC BUFFER
                                        003C   119 ;
                                        003C   120
                              58   D5  003C   121 FILBUF:  TSTL    R8                               ;ANY BUFFER SPECIFIED?
                              43   13  003E   122          BEQL    50$                              ;IF EQL NO
                                        0040   123 ;        IFNORD  #8,(R8),60$                      ;CAN OUTPUT BUFFER DESCRIPTOR BE READ?
                         52   68   3C  0040   124          MOVZWL  (R8),R2                          ;GET SIZE OF OUTPUT BUFFER
                              31   13  0043   125          BEQL    30$                              ;IF EQL NULL
                      53   04 A8   D0  0045   126          MOVL    4(R8),R3                         ;GET ADDRESS OF OUTPUT BUFFER
                                        0049   127 ;        IFNOWRT R2,(R3),60$                      ;CAN ENTIRE BUFFER BE WRITTEN?
                      51   34 A4   DE  0049   128          MOVAL   UCB$L_DEVCHAR(R4),R1             ;GET ADDRESS OF CHARACTERISTICS
                         50   0C   D0  004D   129          MOVL    #12,R0                           ;SET BYTE COUNT
                              16   10  0050   130          BSBB    10$                              ;MOVE CHARACTERISTICS TO BUFFER
                      51   48 A4   DE  0052   131          MOVAL   UCB$W_UNIT(R4),R1                ;GET ADDRESS OF UNIT NUMBER
                         50   02   D0  0056   132          MOVL    #2,R0                            ;SET BYTE COUNT
                              0D   10  0059   133          BSBB    10$                              ;MOVE UNIT NUMBER OT BUFFER
                      51   24 A4   D0  005B   134          MOVL    UCB$L_DDB(R4),R1                 ;GET ADDRESS OF DDB
                      51   14 A1   DE  005F   135          MOVAL   DDB$T_NAME(R1),R1                ;GET ADDRESS OF GENERIC DEVICE NAME
                         50   61   9A  0063   136          MOVZBL  (R1),R0                          ;GET LENGTH OF NAME IN BYTES
                              50   D6  0066   137          INCL    R0                               ;ACCOUNT FOR COUNT BYTE
                              52   D7  0068   138 10$:      DECL    R2                               ;ANY SPACE LEFT IN BUFFER?
                              08   19  006A   139          BLSS    20$                              ;IF LSS NO
                         83   81   90  006C   140          MOVB    (R1)+,(R3)+                      ;MOVE BYTE TO BUFFER
                            F6 50   F5  006F   141          SOBGTR  R0,10$                           ;ANY MORE BYTES TO MOVE?
                              07   11  0072   142          BRB     40$                              ;
                              52   D6  0074   143 20$:      INCL    R2                               ;ADJUST COUNT OF REMAINING BYTES
                50    0601 8F   3C  0076   144 30$:      MOVZWL  #SS$_BUFFEROVF,R0                ;SET BUFFER OVERFLOW
                              57   D5  007B   145 40$:      TSTL    R7                               ;ADDRESS SPECIFIED TO STORE RESULT LENGTH?
                              04   13  007D   146          BEQL    50$                              ;IF EQL NO
                                        007F   147 ;        IFNOWRT #2,(R7),60$                      ;CAN LENGTH BE WRITTEN?
                67    68    52   A3  007F   148          SUBW3   R2,(R8),(R7)                     ;CALCULATE LENGTH OF RESULT STRING
                              05   05  0083   149 50$:      RSB                                      ;
                         50   0C   3C  0084   150 60$:      MOVZWL  #SS$_ACCVIO,R0                   ;SET ACCESS VIOLATION
                              04   04  0087   151          RET                                      ;
                                        0088   152
                                        0088   153          .END
```

CCB$B_AMOD		00000009	D				UCB$B_TT_DETYPE		000000A4	D	
CCB$B_STS		00000008	D				UCB$B_TT_LFFILL		0000009E	D	
CCB$C_LENGTH		00000010	D				UCB$B_TT_SPEED		0000009C	D	
CCB$K_LENGTH		00000010	D				UCB$B_TYPE		0000000A	D	
CCB$L_DIRP		0000000C	D				UCB$B_VERTSZ		0000003F	D	
CCB$L_UCB		00000000	D				UCB$C_LENGTH		00000074	D	
CCB$L_WIND		00000004	D				UCB$C_MB_LENGTH		00000090	D	
CCB$W_IOC		0000000A	D				UCB$C_TT_LENGTH		000000BC	D	
CHAN	=	00000004	D				UCB$K_LENGTH		00000074	D	
DDB$B_ACPCLASS		00000013	D				UCB$K_MB_LENGTH		00000090	D	
DDB$B_TYPE		0000000A	D				UCB$K_TT_LENGTH		000000BC	D	
DDB$C_LENGTH		00000034	D				UCB$L_AMB		00000054	D	
DDB$K_LENGTH		00000034	D				UCB$L_ASTQBL		00000010	D	
DDB$L_ACPD		00000010	D				UCB$L_ASTQFL		0000000C	D	
DDB$L_DDT		0000000C	D				UCB$L_CPID		0000005C	D	
DDB$L_LINK		00000000	D				UCB$L_CRB		00000020	D	
DDB$L_UCB		00000004	D				UCB$L_DDB		00000024	D	
DDB$T_DRVNAME		00000024	D				UCB$L_DEVCHAR		00000034	D	
DDB$T_NAME		00000014	D				UCB$L_DEVDEPEND		0000003C	D	
DDB$W_SIZE		00000008	D				UCB$L_DPC		00000080	D	
DEV$V_SPL	=	00000006	D				UCB$L_DUETIM		0000005C	D	
EXE$GTCHAN		00000000	RG	D	02		UCB$L_DX_BFPNT		0000009C	D	
FILBUF		0000003C	R	D	02		UCB$L_DX_BUF		00000098	D	
IOC$VERIFYCHAN		********		X	02		UCB$L_DX_RXDB		000000A0	D	
PRIBUF	=	0000000C		D			UCB$L_EMB		00000078	D	
PRILEN	=	00000008		D			UCB$L_FIRST		00000014	D	
SCDBUF	=	00000014		D			UCB$L_FPC		0000000C	D	
SCDLEN	=	00000010		D			UCB$L_FQBL		00000004	D	
SIZ...	=	00000002		D			UCB$L_FQFL		00000000	D	
SS$_ACCVIO	=	0000000C		D			UCB$L_FR3		00000010	D	
SS$_BUFFEROVF	=	00000601		D			UCB$L_FR4		00000014	D	
SS$_NORMAL	=	00000001		D			UCB$L_IOQBL		00000044	D	
UCB$B_AMOD		00000053		D			UCB$L_IOQFL		00000040	D	
UCB$B_CEX		00000077		D			UCB$L_IRP		0000004C	D	
UCB$B_CM1		0000004A		D			UCB$L_LINK		0000002C	D	
UCB$B_CM2		0000004B		D			UCB$L_LOGADR		00000064	D	
UCB$B_DEVCLASS		00000038		D			UCB$L_MAXBLOCK		00000084	D	
UCB$B_DEVTYPE		00000039		D			UCB$L_MB_MBX		0000007C	D	
UCB$B_DIPL		00000052		D			UCB$L_MB_PORT		0000008C	D	
UCB$B_DX_SCTCNT		000000A6		D			UCB$L_MB_RAST		00000078	D	
UCB$B_ERTCNT		00000070		D			UCB$L_MB_SHB		00000080	D	
UCB$B_ERTMAX		00000071		D			UCB$L_MB_WAST		00000074	D	
UCB$B_FEX		00000076		D			UCB$L_MB_WIOQBL		00000088	D	
UCB$B_FIPL		0000000B		D			UCB$L_MB_WIOQFL		00000084	D	
UCB$B_LOCSRV		0000003C		D			UCB$L_MEDIA		0000008C	D	
UCB$B_OFFNDX		00000094		D			UCB$L_NT_DATSSB		00000074	D	
UCB$B_OFFRTC		00000095		D			UCB$L_NT_INTSSB		00000078	D	
UCB$B_REMSRV		0000003D		D			UCB$L_OPCNT		00000060	D	
UCB$B_SECTORS		0000003C		D			UCB$L_OWNUIC		0000001C	D	
UCB$B_SLAVE		00000074		D			UCB$L_PID		00000028	D	
UCB$B_SPR		00000075		D			UCB$L_RQBL		00000004	D	
UCB$B_STATE		00000052		D			UCB$L_RQFL		00000000	D	
UCB$B_TRACKS		0000003D		D			UCB$L_SVAPTE		00000068	D	
UCB$B_TT_CRFILL		0000009D		D			UCB$L_SVPN		00000064	D	
UCB$B_TT_DECRF		000000A1		D			UCB$L_TT_DECHAR		000000A8	D	
UCB$B_TT_DELFF		000000A2		D			UCB$L_TT_RDUE		0000008C	D	
UCB$B_TT_DESPEE		000000A0		D			UCB$L_TT_RTIMOU		000000B8	D	

UCB$L_VCB	00000030	D
UCB$T_PARTNER	0000000C	D
UCB$W_BCNT	0000006E	D
UCB$W_BCR	00000096	D
UCB$W_BOFF	0000006C	D
UCB$W_BUFQUO	00000018	D
UCB$W_BYTESTOGO	0000003E	D
UCB$W_CHARGE	0000004A	D
UCB$W_CYLINDERS	0000003E	D
UCB$W_DA	0000008C	D
UCB$W_DC	0000008E	D
UCB$W_DEVBUFSIZ	0000003A	D
UCB$W_DEVSTS	0000005A	D
UCB$W_DIRSEQ	00000088	D
UCB$W_DSTADDR	00000018	D
UCB$W_DX_BCR	000000A4	D
UCB$W_EC1	00000090	D
UCB$W_EC2	00000092	D
UCB$W_ERRCNT	00000072	D
UCB$W_FUNC	0000007E	D
UCB$W_MB_SEED	00000000	D
UCB$W_MSGCNT	00000016	D
UCB$W_MSGMAX	00000014	D
UCB$W_NT_CHAN	00000070	D
UCB$W_OFFSET	0000008A	D
UCB$W_REFC	00000050	D
UCB$W_SIZE	00000008	D
UCB$W_SRCADDR	0000001A	D
UCB$W_STS	00000058	D
UCB$W_TT_DESIZE	000000A5	D
UCB$W_UNIT	00000048	D
UCB$W_VPROT	0000001A	D

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000088 (136.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
CCB$L_UCB	00000000		#-101 (1)
CHAN	=00000004	47 (1)	#-98 (1)
DDB$T_NAME	00000014		135 (1)
DEV$V_SPL	=00000006		#-103 (1) #-110 (1)
EXE$GTCHAN	00000000-R	96 (1)	
FILBUF	0000003C-R	121 (1)	#-107 (1) #-113 (1)
IOC$VERIFYCHAN	00000000-XR		#-99 (1)
PRIBUF	=0000000C	49 (1)	
PRILEN	=00000008	48 (1)	#-106 (1)
SCDBUF	=00000014	51 (1)	
SCDLEN	=00000010	50 (1)	#-112 (1)
SS$_ACCVIO	=0000000C		#-150 (1)
SS$_BUFFEROVF	=00000601		#-144 (1)
SS$_NORMAL	=00000001		#-105 (1)
UCB$L_AMB	00000054		#-104 (1) #-108 (1)
UCB$L_DDB	00000024		#-134 (1)
UCB$L_DEVCHAR	00000034		103 (1) 110 (1) 128 (1)
UCB$W_UNIT	00000048		131 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CCBDEF	1	35 (1)	35 (1)
$DDBDEF	1	36 (1)	36 (1)
$DEFINI	1	35 (1)	35 (1) 36 (1) 37 (1) 38 (1) 39 (1)
$DEVDEF	1	37 (1)	37 (1)
$SSDEF	21	38 (1)	38 (1)
$UCBDEF	10	39 (1)	39 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.09	00:00:00.21
Command processing	110	00:00:00.71	00:00:01.65
Pass 1	349	00:00:08.59	00:00:11.10
Symbol table sort	0	00:00:00.98	00:00:01.14
Pass 2	69	00:00:01.20	00:00:02.86
Symbol table output	18	00:00:00.12	00:00:00.58
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	12	00:00:00.11	00:00:00.16
Assembler run totals	602	00:00:11.83	00:00:17.73

The working set limit was 1000 pages.
41070 bytes (81 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 663 non-local and 10 local symbols.
153 source lines were read in Pass 1, producing 0 object records in Pass 2.
35 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	4
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	10

863 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) GTCHAN/UPDA=(GTCHAN.UPD,GTCHAN.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

```
;    0001  0     %TITLE '*** HELP Handle HELP command'
;    0002  0     MODULE help (                                   ! Supervisor HELP facility
;    0003  0                     IDENT = '06-10'
;    0004  0                     ) =
;    0005  0     !
;    0006  0     ! Copyright (c) 1980, 1982, 1983, 1984
;    0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0008  0     !
;    0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0014  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0015  0     ! REMAIN IN DEC.
;    0016  0     !
;    0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0018  0     ! AND SHC  D  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0019  0     ! CORPORA 'ON.
;    0020  0     !
;    0021  0     ! DEC AS  .S  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
```

```
;    0023  0
;    0024  0     !++
;    0025  0     ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0026  0     !
;    0027  0     ! ABSTRACT:
;    0028  0     !
;    0029  0     !       This module implements a Diagnostic Supervisor HELP command in a
;    0030  0     !       manner more-or-less compatible with VMS.
;    0031  0     !
;    0032  0     ! AUTHOR:       Dave Butenhof   22-jun-1980
;    0033  0     ! MODIFIED BY:
;    0034  0     !
;    0035  0     !               Dave Butenhof 8-sep-1980, version 6.0
;    0036  0     !       1       Convert to use RMS calls for file read.
;    0037  0     !
;    0038  0     !               Dave Butenhof 2-oct-1980, version 6.1
;    0039  0     !       2       Rename entry point to DSX$HELP, since DSV prefix routines
;    0040  0     !               should be JSB entry, whereas DSX is CALLable.
;    0041  0     !               Dave Butenhof, 8-apr-1981, version 6.3
;    0042  0     !       3       On printing other help for not-found key, be sure to skip
;    0043  0     !               the record SAVERFA is pointing to (lest is cause infinite
;    0044  0     !               loop)!
;    0045  0     !              -Dave Butenhof 02-Jun-1981, version 6.4
;    0046  0     !       04      Redefine file specs to make library access more flexible
;    0047  0     !
;    0048  0     !       05      - Dave Butenhof, 03-Feb-1982, Version 6.6
;    0049  0     !               Make KEY_EQUAL be global (as alias DSR$KEY_EQUAL) so
;    0050  0     !               Directory can use it.  Also make it CALL linkage to avoid
;    0051  0     !               complications.
;    0052  0     !
;    0053  0     !       [06]    Dave Butenhof, 29-Mar-1982, version 6.6
;    0054  0     !               Fix bug in READ handling of no-attribute.  Don't check
;    0055  0     !               'last two' characters unless they're really there.
;    0056  0     !       07      Dave butenhof, 20-Apr-1982
;    0057  0     !               Change name HLP$V_HELP to HLP$V_HELP_KEY; LIB now openly defines
;    0058  0     !               something called HLP$V_HELP.
;    0059  0     !
;    0060  0     !       08      Jack Stansbury, 16-Dec-1982, Version 6.11
;    0061  0     !               Changed the field width for qualifiers and other key words to
;    0062  0     !               be 16 rather than 8. This improves the Help output some.
;    0063  0     !
;    0064  0     !       09      M. Baggett,     18-Nov-1983     Version 6.13
;    0065  0     !               The restriction on the second character has been removed.
;    0066  0     !
;    0067  0     !       10      Bob Bergazzi    May 31, 1984    Version 7.0
;    0068  0     !               Edit 09 broke HELP EXAMINE, so fix it and also add support
;    0069  0     !               for all new processors.
;    0070  0     !--
;    0071  0
```

```
;    0072  1     BEGIN
;    0073  1     !+
;    0074  1     ! INCLUDE FILES:
;    0075  1     !-
;    0076  1
;    0077  1     LIBRARY '$diag';                    !                                   [04]
;    0078  1
;    0079  1     LIBRARY '$ds';                      !                                   [04]
;    0080  1
;    0081  1     LIBRARY 'sys$library:starlet.l32';
;    0082  1
```

```
;    0083  1      !+
;    0084  1      ! Linkages
;    0085  1      !-
;    0086  1
;    0087  1      LINKAGE
;    0088  1          jsb_2 = JSB (REGISTER = 0, REGISTER = 1),
;    0089  1          jsb_3 = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2);
;    0090  1
```

```
;    0091  1      !+
;    0092  1      ! TABLE OF CONTENTS:
;    0093  1      !-
;    0094  1
;    0095  1      FORWARD ROUTINE
;    0096  1          dsx$help,                          ! Main program
;    0097  1          do_help,                           ! Recursive file search
;    0098  1          is_key_on_line,                    ! Check for key record
;    0099  1          print_keys : NOVALUE,              ! Print keys level 1 - x
;    0100  1          print_text,                        ! Print help text for level
;    0101  1          print_otherhelp,                   ! List all keys at level
;    0102  1          READ,                              ! Read random RFA
;    0103  1          dsr$key_equal,                     ! Compare keys (wildcard, partial match, etc)
;    0104  1          skip_blanks,                       ! Skip spaces/tabs
;    0105  1          scan_word,                         ! Find length of word
;    0106  1          find_char : jsb_2,                 ! Find character on line
;    0107  1          print_crlf : jsb_none NOVALUE;     ! Cause a blank line to magically appear
;    0108  1
```

```
;    0109  1      !+
;    0110  1      ! PSECT definitions:
;    0111  1      !-
;    0112  1      PSECT
;    0113  1          PLIT = data(SHARE, ADDRESSING_MODE (LONG_RELATIVE));
;    0114  1
;    0115  1      PSECT
;    0116  1          GLOBAL = work(NOSHARE, ADDRESSING_MODE (LONG_RELATIVE));
;    0117  1
;    0118  1      PSECT
;    0119  1          CODE = CODE(EXECUTE, SHARE, NOWRITE);
;    0120  1
```

```
;    0121  1      !+
;    0122  1      ! Literals
;    0123  1      !-
;    0124  1
;    0125  1      BIND
;    0126  1          help_help = $ascid ('HELP');            ! Default first level key
;    0127  1
```

```
;    0128  1      !+
;    0129  1      ! OWN storage:
;    0130  1      !-
;    0131  1
;    0132  1      GLOBAL
;    0133  1          helpinfo : bblock [hlp$c_size];          ! Help processing data block
;    0134  1
```

```
;    0135  1     %SBTTL 'Help control routine'
;    0136  1
;    0137  1     GLOBAL ROUTINE dsx$help (desc) =
;    0138  1
;    0139  1     !++
;    0140  1     ! FUNCTIONAL DESCRIPTION
;    0141  1     !       Process HELP command.  Scans appropriate file to extract help text
;    0142  1     !       for keywords
;    0143  1     !
;    0144  1     ! INPUTS
;    0145  1     !       desc    address of descriptor for remainder of command line
;    0146  1     !
;    0147  1     ! IMPLICIT INPUTS
;    0148  1     !       none
;    0149  1     !
;    0150  1     ! OUTPUTS
;    0151  1     !       desc    descriptor is updated to end of line
;    0152  1     !
;    0153  1     ! IMPLICIT OUTPUTS
;    0154  1     !       typeout to console
;    0155  1     !
;    0156  1     ! SIDE EFFECTS
;    0157  1     !       none
;    0158  1     !
;    0159  1     ! RETURN CODES
;    0160  1     !       none
;    0161  1     !--
;    0162  1
```

```
;    0163  2          BEGIN
;    0164  2
;    0165  2          LOCAL
;    0166  2              helpfab : $fab_decl,                        ! FAB block
;    0167  2              helprab : $rab_decl,                        ! RAB block
;    0168  2              startrfa : bblock [rfa$c_length],           ! Beginning of file
;    0169  2              buffer : bblock [hlp$c_maxrecsiz],          ! Allocate input buffer
;    0170  2              nextkey : VECTOR [hlp$c_maxrecsiz, BYTE],
;    0171  2              nextdesc : bblock [dsc$c_s_bln],
;    0172  2              level,
;    0173  2              elipsis,
;    0174  2              keylist : bblock [hlp$c_maxkeys*dsc$c_s_bln];   ! Keyword descriptors
;    0175  2
;    0176  2          MAP
;    0177  2              desc : REF bblock [dsc$c_s_bln];
;    0178  2
;    0179  2          print_crlf ();                                  ! Leading blank line
;    0180  2          elipsis = '....';                               ! Three dots
;    0181  2          CH$FILL (0, hlp$c_size, helpinfo);              ! Clear data table
;    0182  2          helpinfo [hlp$l_allhelp] = hlp$c_maxkeys + 1;       ! Init 'allhelp' to deactivate it
;    0183  2          CH$FILL (0, hlp$c_maxkeys*dsc$c_s_bln, keylist);    ! Clear key descriptors
;    0184  2          helpinfo [hlp$l_keylist] = keylist;             ! Set up pointer to keys
;    0185  2          level = 0;
;    0186  2          helpinfo [hlp$l_rab] = helprab;                 ! Set up pointer to RMS
;    0187  2          $fab_init (fab = helpfab, dnm = '.HLP', fnm = 'EVSAA');     ! Initialize FAB
;    0188  2          $rab_init (rab = helprab, fab = helpfab, ubf = buffer, usz = hlp$c_maxrecsiz);      ! Initialize RAB
;    0189  2
;    0190  2          WHILE 1 DO                                      ! Isolate the keywords
;    0191  3              BEGIN
;    0192  3
;    0193  3              BIND
;    0194  3                  curkey = keylist + dsc$c_s_bln*.level : bblock,
;    0195  3                  wildflags = helpinfo [hlp$b_wildflags] : BITVECTOR,
;    0196  3                  keytext = curkey [dsc$a_pointer] : REF VECTOR [, BYTE];
;    0197  3
;    0198  3              LOCAL
;    0199  3                  next_qual,
;    0200  3                  ptr;
;    0201  3
;    0202  3              IF NOT skip_blanks (.desc) THEN EXITLOOP;       ! End of line
;    0203  3
;    0204  3              curkey [dsc$a_pointer] = .desc [dsc$a_pointer]; ! Beginning of a word
;    0205  3              next_qual = find_char (.desc, %C'/');
;    0206  3
;    0207  3              IF .next_qual EQL 0                         ! If we're lookin' at it,
;    0208  3              THEN
;    0209  3                  next_qual = .desc [dsc$w_length];       !  it's a start, not an end
;    0210  3
;    0211  3              curkey [dsc$w_length] =                     ! Find length of word
;    0212  3              MIN (.next_qual, find_char (.desc, %C' '), find_char (.desc, %C'        '));
;    0213  3              desc [dsc$w_length] = .desc [dsc$w_length] -    ! Update descriptor
;    0214  3              .curkey [dsc$w_length];
;    0215  3              desc [dsc$a_pointer] = .desc [dsc$a_pointer] + .curkey [dsc$w_length];
;    0216  3              ptr = CH$FIND_SUB (.curkey [dsc$w_length], .curkey [dsc$a_pointer], 3, elipsis);
;    0217  3                                                          ! Look for '...' in key
;    0218  3
;    0219  4              IF NOT CH$FAIL (.ptr) AND .ptr EQL (.curkey [dsc$a_pointer] + .curkey [dsc$w_length] - 3)
```

```
;    0220  3              THEN                                        ! "<word>..."
;    0221  4                  BEGIN
;    0222  4                  helpinfo [hlp$l_allhelp] = .level + 2;      ! All lower keys wildcarded
;    0223  4                  curkey [dsc$w_length] = .curkey [dsc$w_length] - 3; ! Discard '...'
;    0224  4
;    0225  4                  IF .curkey [dsc$w_length] EQL 0     ! If just '...', wilcard this level, too
;    0226  4                  THEN
;    0227  4                      helpinfo [hlp$l_allhelp] = .helpinfo [hlp$l_allhelp] - 1
;    0228  4                  ELSE
;    0229  4                      level = .level + 1;             ! Count level if '...' had company
;    0230  4
;    0231  4                  INCR i FROM .helpinfo [hlp$l_allhelp] TO hlp$c_maxkeys DO
;    0232  4                      wildflags [.i - 1] = 1;         ! Set the wildcard flags
;    0233  4
;    0234  4                  EXITLOOP                            ! Don't need more keys
;    0235  3                  END;
;    0236  3
;    0237  3              INCR i FROM 0 TO .curkey [dsc$w_length] - 1 DO
;    0238  3
;    0239  3                  IF .keytext [.i] EQL %C'%' OR .keytext [.i] EQL %C'*' THEN wildflags [.level] = 1;
;    0240  3
;    0241  3                                                      ! Set flag if wildcard char.
;    0242  3              level = .level + 1
;    0243  2              END;
;    0244  2
;    0245  2          helpinfo [hlp$l_realkeys] = .level;         ! Number of keys
;    0246  2
;    0247  2          IF .level GTR 0
;    0248  2          THEN                                        ! If there's a key,
;    0249  3              BEGIN                                   ! Check for diagnostic help
;    0250  3
;    0251  3              BIND
;    0252  3                  key1text = keylist [dsc$a_pointer] : REF VECTOR [, BYTE];
;    0253  3
;    0254  4              IF .key1text [0] EQL %C'E' AND (.KEY1TEXT [1] EQL %C'V' OR      !                       [10]
;    0255  4                                              .KEY1TEXT [1] EQL %C'S' OR      !                       [10]
;    0256  4                                              .KEY1TEXT [1] EQL %C'C' OR      !                       [10]
;    0257  4                                              .KEY1TEXT [1] EQL %C'N' OR      !                       [10]
;    0258  4                                              .KEY1TEXT [1] EQL %C'T' OR      !                       [10]
;    0259  4                                              .KEY1TEXT [1] EQL %C'D' OR      !                       [10]
;    0260  4                                              .KEY1TEXT [1] EQL %C'B' OR      !                       [10]
;    0261  4                                              .KEY1TEXT [1] EQL %C'M' OR      !                       [10]
;    0262  4                                              .KEY1TEXT [1] EQL %C'Z')        !                       [10]
;    0263  3              THEN
;    0264  4                  BEGIN                               ! Key 1 is actually file specifier for diagnostic help file
;    0265  4                  helpfab [fab$l_fna] = .keylist [dsc$a_pointer];     ! Change filename
;    0266  4                  helpfab [fab$b_fns] = .keylist [dsc$w_length];
;    0267  4                  CH$MOVE ((hlp$c_maxkeys - 1)*dsc$c_s_bln, keylist + dsc$c_s_bln, keylist);
;    0268  4                                                      ! Move other keys down one
;    0269  4                  helpinfo [hlp$b_wildflags] = .helpinfo [hlp$b_wildflags]^-1;        ! Move flags down one bit
;    0270  4
;    0271  4                  IF .helpinfo [hlp$l_allhelp] LEQ hlp$c_maxkeys
;    0272  4                  THEN                                ! Correct ALLHELP level
;    0273  4                      helpinfo [hlp$l_allhelp] = .helpinfo [hlp$l_allhelp] - 1;
;    0274  4
;    0275  4                  helpinfo [hlp$l_realkeys] = .helpinfo [hlp$l_realkeys] - 1
;    0276  4                  END
```

```
;    0277  4
;    0278  2              END;
;    0279  2
;    0280  2          IF .helpinfo [hlp$l_realkeys] EQL 0 OR .keylist [dsc$w_length] EQL 0
;    0281  2          THEN                                        ! If null first key, make it
;    0282  3              BEGIN                                   ! 'HELP' by default
;    0283  3              CH$MOVE (dsc$c_s_bln, help_help, keylist);
;    0284  3              helpinfo [hlp$l_realkeys] = 1           ! Set one key found
;    0285  2              END;
;    0286  2
;    0287  2          IF NOT .(helpinfo [hlp$b_wildflags])<0, 1, 0>       !  If key 1 not wild
;    0288  2          THEN
;    0289  2
;    0290  3              IF (
;    0291  3
;    0292  3                  LOCAL
;    0293  3                      w;
;    0294  3
;    0295  3                  dsr$key_equal (help_help, keylist, w))      !  and it matches 'HELP'
;    0296  2              THEN
;    0297  2                  helpinfo [hlp$v_help_key] = 1;      !  then set flag                    [07]
;    0298  2
;    0299  2          startrfa [rfa$l_vbn] = 1;                   ! Start of file
;    0300  2          startrfa [rfa$w_offset] = 0;
;    0301  3          (                                           ! Open the file
;    0302  3
;    0303  3          LOCAL
;    0304  3              stat;
;    0305  3
;    0306  3          stat = $open (fab = helpfab); IF NOT .stat THEN RETURN .stat);
;    0307  3
;    0308  3          IF .helpfab [fab$b_rat] NEQ 0               ! If not no-attribute
;    0309  3          THEN
;    0310  3              helpinfo [hlp$v_cr] = 1;                ! Need explicit CR on typeout
;    0311  3
;    0312  4          (
;    0313  4
;    0314  4          LOCAL
;    0315  4              stat;
;    0316  4
;    0317  4          stat = $connect (rab = helprab);            ! Connect record stream
;    0318  4          IF NOT .stat THEN RETURN .stat);
;    0319  4          nextdesc [dsc$a_pointer] = nextkey;         ! Set key space for key 1
;    0320  4          (
;    0321  4
;    0322  4          LOCAL
;    0323  4              stat;
;    0324  4
;    0325  4          stat = do_help (1, startrfa, nextdesc);     ! Do everything, recursively
;    0326  4          IF NOT .stat THEN RETURN .stat);
;    0327  4          BEGIN                                       ! Check for tail-end stuff
;    0328  4
;    0329  4          LOCAL
;    0330  4              noerr;
;    0331  4
;    0332  4          startrfa [rfa$l_vbn] = 1;                   ! Reset RFA
;    0333  4          startrfa [rfa$w_offset] = 0;
```

```
;    0334  3
;    0335  4          IF NOT (noerr = .helpinfo [hlp$v_foundhel; 1)          ! If no help found
;    0336  3              OR .helpinfo [hlp$v_help_key]                      !  or key 1 was 'HELP'                              [07]
;    0337  3          THEN
;    0338  4              (
;    0339  4
;    0340  4              LOCAL
;    0341  4                  stat;
;    0342  4
;    0343  4              stat = print_otherhelp (1, startrfa, .noerr);      ! print error and/or otherhelp
;    0344  4              IF NOT .stat THEN RETURN .stat)
;    0345  4
;    0346  2          END;
;    0347  2          print_crlf ();                                         ! Cause a trailing CRLF
;    0348  3          $close (fab = helpfab)                                 ! Close the file
;    0349  1          END;                                                   ! of help


                                                                                .TITLE  HELP *** HELP Handle HELP command
                                                                                .IDENT  \06-10\

                                                                                .PSECT  WORK,NOEXE,2

                                                                00000 HELPINFO::
                                                                                .BLKB   46

                                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                                 50  4C  45  48   00000 P.AAB:   .ASCII  \HELP\                                  ;
                                                       00000004    00004 P.AAA:   .LONG   4                                       ;
                                                       00000000'   00008          .ADDRESS P.AAB                              ;
                                               41  41  53  56  45  0000C P.AAC:   .ASCII  \EVSAA\                                ;
                                                   50  4C  48  2E  00011 P.AAD:   .ASCII  \.HLP\                                ;

                                                                               HELP_HELP=          P.AAA
                                                                               WILDFLAGS=          HELPINFO+44
                                                                                       .EXTRN  SYS$OPEN, SYS$CONNECT
                                                                                       .EXTRN  SYS$CLOSE

                                                                                       .PSECT  CODE,NOWRT,  SHR,2

                                                                00FFC 00000            .ENTRY  DSX$HELP, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0137
                                                                                               R11                                 ;
                                                 59     0000V   CF  9E  00002          MOVAB   FIND_CHAR, R9                       ;
                                                 58 00000000'   EF  9E  00007          MOVAB   HELP_HELP, R8                       ;
                                                 57 00000000'   EF  9E  0000E          MOVAB   HELPINFO, R7                        ;
                                                 5E     FD2C    CE  9E  00015          MOVAB   -724(SP), SP                        ;
                                                          0000V 30  00  0001A          BSBW    PRINT_CRLF                          ; 0179
                                                 6E 2E2E2E2E    8F  D0  0001D          MOVL    #774778414, ELIPSIS                 ; 0180
                                         2E      6E             00  2C  00024          MOVC5   #0, (SP), #0, #46, HELPINFO         ; 0181
                                                                67      00029                                                      ;
                                                 67             06  D0  0002A          MOVL    #6, HELPINFO                        ; 0182
                                         28      6E             00  2C  0002D          MOVC5   #0, (SP), #0, #40, KEYLIST          ; 0183
                                                          08    AE      00032                                                      ;
                                             24  A7       08    AE  9E  00034          MOVAB   KEYLIST, HELPINFO+36                ; 0184
                                                                56  D4  00039          CLRL    LEVEL                               ; 0185
                                             04  A7     FF6C    CD  9E  0003B          MOVAB   HELPRAB, HELPINFO+4                 ; 0186
```

```
   0050  8F        00           6E          00  2C 00041        MOVC5   #0, (SP), #0, #80, $RMS_PTR                ; 0187
                                       B0   AD     00048                                                           ;
                           B0   AD   5003   8F  B0 0004A        MOVW    #20483, $RMS_PTR                           ;
                           C6   AD     02   90  00050           MOVB    #2, $RMS_PTR+22                            ;
                           CF   AD     02   90  00054           MOVB    #2, $RMS_PTR+31                            ;
                           DC   AD     08   A8  9E 00058        MOVAB   P.AAC, $RMS_PTR+44                         ;
                           E0   AD     0D   A8  9E 0005D        MOVAB   P.AAD, $RMS_PTR+48                         ;
                           E4   AD   0405   8F  B0 00062        MOVW    #1029, $RMS_PTR+52                         ;
   0044  8F        00           6E          00  2C 00068        MOVC5   #0, (SP), #0, #68, $RMS_PTR                ; 0188
                                     FF6C   CD     0006F                                                           ;
                         FF6C   CD   4401   8F  D0 00072        MOVW    #17409, $RMS_PTR                           ;
                           8C   AD   0100   8F  B0 00079        MOVW    #256, $RMS_PTR+32                          ;
                           90   AD   0138   CE  9E 0007F        MOVAB   BUFFER, $RMS_PTR+36                        ;
                           A8   AD     B0   AD  9E 00085        MOVAB   HELPFAB, $RMS_PTR+60                       ;
                                55     04   AC  D0 0009A        MOVL    DESC, R5                                   ; 0202
                                54     08 AE46  7E 0008E 1$:    MOVAQ   KEYLIST[LEVEL], R4                         ; 0194
                                            55  DD 00093        PUSHL   R5                                         ; 0202
                        0000V   CF          01  FB 00095        CALLS   #1, SKIP_BLANKS                            ;
                                03          50  E8 0009A        BLBS    R0, 2$                                     ;
                                          00A9  31 0009D        BRW     15$                                        ;
                           04   A4     04   A5  D0 000A0 2$:    MOVL    4(R5), 4(R4)                               ; 0204
                                51          2F  D0 000A5        MOVL    #47, R1                                    ; 0205
                                50          55  D0 000A8        MOVL    R5, R0                                     ;
                                            69  16 000AB        JSB     FIND_CHAR                                  ;
                                52          50  D0 000AD        MOVL    R0, NEXT_QUAL                              ;
                                            03  12 000B0        BNEQ    3$                                         ; 0207
                                52          65  3C 000B2        MOVZWL  (R5), NEXT_QUAL                            ; 0209
                                51          20  D0 000B5 3$:    MOVL    #32, R1                                    ; 0212
                                50          55  D0 000B8        MOVL    R5, R0                                     ;
                                            69  16 000BB        JSB     FIND_CHAR                                  ;
                                50          52  D1 000BD        CMPL    R2, R0                                     ;
                                            03  15 000C0        BLEQ    4$                                         ;
                                52          50  D0 000C2        MOVL    R0, R2                                     ;
                                51          09  D0 000C5 4$:    MOVL    #9, R1                                     ;
                                50          55  D0 000C8        MOVL    R5, R0                                     ;
                                            69  16 000CB        JSB     FIND_CHAR                                  ;
                                50          52  D1 000CD        CMPL    R2, R0                                     ;
                                            03  15 000D0        BLEQ    5$                                         ;
                                52          50  D0 000D2        MOVL    R0, R2                                     ;
                                64          52  B0 000D5 5$:    MOVW    R2, (R4)                                   ;
                                65          64  A2 000D8        SUBW2   (R4), (R5)                                 ; 0214
                                50          64  3C 000DB        MOVZWL  (R4), R0                                   ; 0215
                           04   A5          50  C0 000DE        ADDL2   R0, 4(R5)                                  ;
     04  B4        64           6E          03  39 000E2        MATCHC  #3, ELIPSIS, (R4), @4(R4)                  ; 0216
                                            03  13 000E8        BEQL    6$                                         ;
                                53          03  D0 000EA        MOVL    #3, R3                                     ;
                                53          03  C2 000ED 6$:    SUBL2   #3, R3                                     ;
                                            33  13 000F0        BEQL    11$                                        ; 0219
                                50          64  3C 000F2        MOVZWL  (R4), R0                                   ;
                                50     04   A4  C0 000F5        ADDL2   4(R4), R0                                  ;
                                50          03  C2 000F9        SUBL2   #3, R0                                     ;
                                50          53  D1 000FC        CMPL    PTR, R0                                    ;
                                            24  12 000FF        BNEQ    11$                                        ;
                                67     02   A6  9E 00101        MOVAB   2(R6), HELPINFO                            ; 0222
                                64          03  A2 00105        SUBW2   #3, (R4)                                   ; 0223
                                            04  12 00108        BNEQ    7$                                         ; 0225
                                            67  D7 0010A        DECL    HELPINFO                                   ; 0227
```

```
                                                02  11  0010C          BRB     8$                                          ;
                                                56  D6  0010E 7$:      INCL    LEVEL                                       ; 0229
                50              67              01  C3  00110 8$:      SUBL3   #1, HELPINFO, I                             ; 0231
                                                09  11  00114          BRB     10$                                         ;
                                51          FF  A0  9E  00116 9$:      MOVAB   -1(R0), R1                                  ; 0232
                00          2C  A7              51  E2  0011A          BBSS    R1, WILDFLAGS, 10$                          ;
                F3              50              05  F3  0011F 10$:     AOBLEQ  #5, I, 9$                                   ;
                                                24  11  00123          BRB     15$                                         ; 0221
                                51              64  3C  00125 11$:     MOVZWL  (R4), R1                                    ; 0237
                                50              01  CE  00128          MNEGL   #1, I                                       ;
                                                13  11  0012B          BRB     14$                                         ;
                                25          04  B440    91  0012D 12$:     CMPB    @4(R4)[I], #37                      ; 0239
                                                07  13  00132          BEQL    13$                                         ;
                                2A          04  B440    91  00134          CMPB    @4(R4)[I], #42                      ;
                                                05  12  00139          BNEQ    14$                                         ;
                00          2C  A7              56  E2  0013B 13$:     BBSS    LEVEL, WILDFLAGS, 14$                       ;
                E9              50              51  F2  00140 14$:     AOBLSS  R1, I, 12$                                  ;
                                                56  D6  00144          INCL    LEVEL                                       ; 0242
                                            FF45    31  00146          BRW     1$                                          ;
                            28  A7              56  D0  00149 15$:     MOVL    LEVEL, HELPINFO+40                          ; 0245
                                                6A  15  0014D          BLEQ    18$                                         ; 0247
                                51          0C  AE  D0  0014F          MOVL    KEY1TEXT, R1                                ; 0254
                            45  8F              61  91  00153          CMPB    (R1), #69                                   ;
                                                60  12  00157          BNEQ    18$                                         ;
                                50          01  A1  9A  00159          MOVZBL  1(R1), R0                                   ;
                            56  8F              50  91  0015D          CMPB    R0, #86                                     ;
                                                30  13  00161          BEQL    16$                                         ;
                            53  8F              50  91  00163          CMPB    R0, #83                                     ; 0255
                                                2A  13  00167          BEQL    16$                                         ;
                            43  8F              50  91  00169          CMPB    R0, #67                                     ; 0256
                                                24  13  0016D          BEQL    16$                                         ;
                            4E  8F              50  91  0016F          CMPB    R0, #78                                     ; 0257
                                                1E  13  00173          BEQL    16$                                         ;
                            54  8F              50  91  00175          CMPB    R0, #84                                     ; 0258
                                                18  13  00179          BEQL    16$                                         ;
                            44  8F              50  91  0017B          CMPB    R0, #68                                     ; 0259
                                                12  13  0017F          BEQL    16$                                         ;
                            42  8F              50  91  00181          CMPB    R0, #66                                     ; 0260
                                                0C  13  00185          BEQL    16$                                         ;
                            4D  8F              50  91  00187          CMPB    R0, #77                                     ; 0261
                                                06  13  0018B          BEQL    16$                                         ;
                            5A  8F              50  91  0018D          CMPB    R0, #90                                     ; 0262
                                                26  12  00191          BNEQ    18$                                         ;
                            DC  AD              51  D0  00193 16$:     MOVL    R1, HELPFAB+44                              ; 0265
                            E4  AD          08  AE  90  00197          MOVB    KEYLIST, HELPFAB+52                         ; 0266
    08  AE              10  AE              20  28  0019C          MOVC3   #32, KEYLIST+8, KEYLIST                     ; 0267
                                50          2C  A7  9A  001A2          MOVZBL  HELPINFO+44, R0                             ; 0269
        50                      50          FF  8F  78  001A6          ASHL    #-1, R0, R0                                 ;
                            2C  A7              50  90  001AB          MOVB    R0, HELPINFO+44                             ;
                                05              67  D1  001AF          CMPL    HELPINFO, #5                                ; 0271
                                                02  14  001B2          BGTR    17$                                         ;
                                                67  D7  001B4          DECL    HELPINFO                                    ; 0273
                                            28  A7  D7  001B6 17$:     DECL    HELPINFO+40                                 ; 0275
                                            28  A7  D5  001B9 18$:     TSTL    HELPINFO+40                                 ; 0280
                                                05  13  001BC          BEQL    19$                                         ;
                                            08  AE  B5  001BE          TSTW    KEYLIST                                     ;
                                                09  12  001C1          BNEQ    20$                                         ;
```

```
                08  AE          68                  08  28 001C3 19$:   MOVC3   #8, HELP_HELP, KEYLIST                  ; 0283
                          28    A7                  01  D0 001C8       MOVL    #1, HELPINFO+40                         ; 0284
                                14                  2C  A7 E8 001CC 20$: BLBS    HELPINFO+44, 21$                    ; 0287
                                                    04  AE 9F 001D0     PUSHAB  W                                       ; 0295
                                                    0C  AE 9F 001D3     PUSHAB  KEYLIST                                 ;
                                                    58  DD 001D6       PUSHL   R8                                      ;
                            0000V   CF              03  FB 001D8       CALLS   #3, DSR$KEY_EQUAL                       ;
                                    04              50  E9 001DD       BLBC    R0, 21$                                 ;
                                2D  A7              01  88 001E0       BISB2   #1, HELPINFO+45                         ; 0297
                              FF64  CD              01  D0 001E4 21$:   MOVL    #1, STARTRFA                            ; 0299
                                                FF68  CD  B4 001E9       CLRW    STARTRFA+4                              ; 0300
                                                    B0  AD 9F 001ED     PUSHAB  HELPFAB                                 ; 0306
                        00000000G   00              01  FB 001F0       CALLS   #1, SYS$OPEN                            ;
                                    60              50  E9 001F7       BLBC    STAT, 25$                               ;
                                                    CE  AD 95 001FA     TSTB    HELPFAB+30                              ; 0308
                                                    04  13 001FD       BEQL    22$                                     ;
                                2D  A7              08  88 001FF       BISB2   #8, HELPINFO+45                         ; 0310
                                                FF6C  CD  9F 00203 22$: PUSHAB  HELPRAB                                 ; 0317
                        00000000G   00              01  FB 00207       CALLS   #1, SYS$CONNECT                         ;
                                    49              50  E9 0020E       BLBC    STAT, 25$                               ; 0318
                                34  AE          38  AE 9E 00211       MOVAB   NEXTKEY, NEXTDESC+4                     ; 0319
                                                    30  AE 9F 00216     PUSHAB  NEXTDESC                                ; 0325
                                                FF64  CD  9F 00219     PUSHAB  STARTRFA                                ;
                                                    01  DD 0021D       PUSHL   #1                                      ;
                            0000V   CF              03  FB 0021F       CALLS   #3, DO_HELP                             ;
                                    33              50  E9 00224       BLBC    STAT, 25$                               ; 0326
                              FF64  CD              01  D0 00227       MOVL    #1, STARTRFA                            ; 0332
                                                FF68  CD  B4 0022C       CLRW    STARTRFA+4                              ; 0333
          50        2D  A7              01          02  EF 00230       EXTZV   #2, #1, HELPINFO+45, NOERR              ; 0335
                                        04          50  E9 00236       BLBC    NOERR, 23$                              ;
                                        10      2D  A7  E9 00239       BLBC    HELPINFO+45, 24$                        ; 0336
                                                    50  DD 0023D 23$:   PUSHL   NOERR                                  ; 0343
                                                FF64  CD  9F 0023F     PUSHAB  STARTRFA                                ;
                                                    01  DD 00243       PUSHL   #1                                      ;
                            0000V   CF              03  FB 00245       CALLS   #3, PRINT_OTHERHELP                     ;
                                    0D              50  E9 0024A       BLBC    STAT, 25$                               ; 0344
                                                0000V   30  00240 24$: BSBW    PRINT_CRLF                              ; 0347
                                                    B0  AD 9F 00250     PUSHAB  HELPFAB                                 ; 0348
                        00000000G   00              01  FB 00253       CALLS   #1, SYS$CLOSE                           ;
                                                        04  0025A 25$: RET                                             ; 0349

; Routine Size:  603 bytes,    Routine Base:  CODE + 0000

;   0350  1
```

```
;    0351  1      %SBTTL 'do help stuff'
;    0352  1      ROUTINE do_help (level, firstrfa, key) =
;    0353  1
;    0354  1      !++
;    0355  1      ! FUNCTIONAL DESCRIPTION
;    0356  1      !       Recursive routine to scan for all help.  Each incarnation will call
;    0357  1      !       itself at each matched key of it's level, to scan lower levels of
;    0358  1      !       that key.  Matched keys are printed, lowest level help text is
;    0359  1      !       printed.  If there is no match, the message 'No help for...' is
;    0360  1      !       typed, and a list of other help available is typed.  This list is
;    0361  1      !       typed anyway at the first level if the first level key is 'HELP'.
;    0362  1      !
;    0363  1      ! INPUTS
;    0364  1      !       level           the level of help for which this incarnation is
;    0365  1      !                       to extract help.
;    0366  1      !
;    0367  1      !       firstrfa        The RFA to begin searching the file at.
;    0368  1      !       key             address of descriptor to save our key
;    0369  1      !
;    0370  1      ! IMPLICIT INPUTS
;    0371  1      !       none
;    0372  1      !
;    0373  1      ! OUTPUTS
;    0374  1      !       none
;    0375  1      !
;    0376  1      ! IMPLICIT OUTPUTS
;    0377  1      !       several flags in the HELPINFO block.
;    0378  1      !
;    0379  1      ! SIDE EFFECTS
;    0380  1      !       updates RFA block beyond text used at this level, unless Allhelped
;    0381  1      !
;    0382  1      ! RETURN CODES
;    0383  1      !       none
;    0384  1      !--
;    0385  1
```

```
;    0386  2          BEGIN
;    0387  2
;    0388  2          BIND
;    0389  2              rab = .helpinfo [hlp$l_rab] : bblock,
;    0390  2              curkey = (.helpinfo [hlp$l_keylist] + ((.level - 1)*dsc$c_s_bln)) : bblock;
;    0391  2
;    0392  2          MAP
;    0393  2              key : REF bblock,
;    0394  2              firstrfa : REF bblock;
;    0395  2
;    0396  2          LOCAL
;    0397  2              linelevel,                              ! level of current key line
;    0398  2              qualifier,                              ! Set if curkey begins w/ "/"
;    0399  2              char : BYTE,
;    0400  2              nextkey : VECTOR [hlp$c_maxrecsiz, BYTE],       ! space to hold lower key
;    0401  2              nextdesc : bblock [dsc$c_s_bln],        ! descriptor lower key
;    0402  2              keydesc : bblock [dsc$c_s_bln],
;    0403  2              bound_level,
;    0404  2              lastlevel,                              ! last key level found
;    0405  2              saverfa : bblock [rfa$c_length];        ! Where we started
;    0406  2
;    0407  3          IF .level GTR hlp$c_maxkeys OR (.level GTR .helpinfo [hlp$l_realkeys] AND .level LSS .helpinfo [hlp$l_allhelp
;    0408  3              ])
;    0409  2          THEN
;    0410  2              RETURN 1;                               ! We oughtn't to be here,
;    0411  2
;    0412  2          IF .curkey [dsc$a_pointer] NEQ 0            ! If key is not null
;    0413  2          THEN
;    0414  2              char = .(.curkey [dsc$a_pointer])<0, 8, 0>      ! Get first char of input key
;    0415  2          ELSE
;    0416  2              char = 0;                               ! otherwise, clear it
;    0417  2
;    0418  2          qualifier = .char EQL %C'/';
;    0419  2          bound_level = lastlevel = .level - 1;       ! Init last level found
;    0420  2          CH$MOVE (rfa$c_length, .firstrfa, rab [rab$w_rfa]); ! Set RFA to read
;    0421  2
;    0422  2          DO
;    0423  3              BEGIN                                   ! Find first key of right level
;    0424  3
;    0425  3              LOCAL
;    0426  3                  stat;                               ! status
;    0427  3
;    0428  3              CH$MOVE (rfa$c_length, rab [rab$w_rfa], saverfa);       ! Save start RFA
;    0429  3              stat = READ ();                         ! Read a record
;    043C  3
;    043; 3               IF NOT .stat                            ! If some nasty error hit us
;    0432  3              THEN
;    0433  3
;    0434  3                  IF .stat EQL rms$_eof THEN EXITLOOP ELSE RETURN .stat;      ! Say it didn't work out
;    0435  3
;    0436  3              linelevel = is_key_on_line (keydesc);   ! Get level and key from line
;    0437  3              END
;    0438  4          UNTIL (IF NOT .qualifier THEN (.linelevel LEQ .level) ELSE (.linelevel EQL hlp$c_maxkeys + 1 OR .linelevel
;    0439  3              LEQ .bound_level))                      ! Qual. search can skip 1 level
;    0440  2              AND .linelevel NEQ 0;                   ! until we find our level, or won't
;    0441  2
;    0442  2          IF .linelevel LSS .level                    ! No help at this level
```

```
;   0443  2              AND NOT .qualifier
;   0444  2          THEN
;   0445  3              BEGIN
;   0446  3              CH$MOVE (rfa$c_length, rab [rab$w_rfa], .firstrfa);     ! Backup RFA
;   0447  3              RETURN 1
;   0448  2              END;
;   0449  2
```

```
;    0450  2
;    0451  2          !+
;    0452  2          ! Now process the file, from current position to the next key of higher level
;    0453  2          ! (i.e., level 2 processing ends when level 1 key is found, level 1
;    0454  2          ! processing ends when end-of-file is reached).
;    0455  2          !-
;    0456  2
;    0457  2          rab [rab$b_rac] = rab$c_rfa;                ! Set random for fst record
;    0458  2
;    0459  2          WHILE 1 DO
;    0460  3              BEGIN
;    0461  3
;    0462  3              LOCAL
;    0463  3                  wildkey,                            ! Flag for wildcarding key
;    0464  3                  stat,                               ! Status
;    0465  3                  linelevel;                          ! level of current line
;    0466  3
;    0467  3              stat = READ ();                         ! Read next record
;    0468  3
;    0469  3              IF NOT .stat
;    0470  3              THEN
;    0471  3
;    0472  3                  IF .stat EQL rms$_eof THEN EXITLOOP ELSE RETURN .stat;
;    0473  3
;    0474  3              linelevel = is_key_on_line (keydesc);   ! Find key on this line
;    0475  3
;    0476  3              IF .linelevel LEQ hlp$c_maxkeys         ! If this is numeric key
;    0477  3                  AND .linelevel GTR 0                !  level (not qualifier)
;    0478  3              THEN
;    0479  3                  lastlevel = .linelevel;             ! then remember it
;    0480  3
;    0481  3              IF .linelevel LEQ .bound_level AND .linelevel NEQ 0 THEN EXITLOOP;      ! Exit if found key we don't want
;    0482  3
;    0483  3              IF .linelevel EQL .level                ! If levels match,
;    0484  3                  OR .level GEQ .helpinfo [hlp$l_allhelp]     !  or ALLHELPed
;    0485  4                  OR (.qualifier                      !  or key is qualifier
;    0486  4                  AND .linelevel EQL hlp$c_maxkeys + 1        !   and so is line
;    0487  4                  AND .lastlevel EQL .bound_level)    !   and it's at right level
;    0488  3              THEN                                    ! This is a possible
;    0489  3
;    0490  3                  IF dsr$key_equal (keydesc, curkey, wildkey) ! Compare the keys
;    0491  3                      OR .level GEQ .helpinfo [hlp$l_allhelp] !  see if all help ('...') used
;    0492  3                  THEN
;    0493  4                      BEGIN                           ! They ARE equal...go to it
;    0494  4
;    0495  4                      BIND
;    0496  4                          keynames = helpinfo [hlp$l_keynames] : VECTOR [hlp$c_maxkeys, LONG];
;    0497  4
;    0498  4                      keynames [.bound_level] = .key; ! Set up static key pointer
;    0499  4                      CH$MOVE (key [dsc$w_length] =   ! Set up key text--all rest
;    0500  4                          .helpinfo [hlp$l_bufdesc] -         !  or record from key on
;    0501  4                          (.keydesc [dsc$a_pointer] - .(helpinfo [hlp$l_bufdesc] + 4)), .keydesc [dsc$a_pointer],
;    0502  4                          .key [dsc$a_pointer]);      ! Copy key to save it
;    0503  4                      helpinfo [hlp$v_qual] = .linelevel EQL hlp$c_maxkeys + 1;       ! Set flag if this is qualifier
;    0504  4
;    0505  4                      IF .level EQL .helpinfo [hlp$l_realkeys]        ! If this is lowest level
;    0506  4                          OR .level GEQ .helpinfo [hlp$l_allhelp]     !  or we want it all
```

```
;    0507  4                    THEN
;    0508  5                        BEGIN
;    0509  5                        helpinfo [hlp$v_foundhelp] = 1;     ! We've found something
;    0510  5                        print_keys (.level);            ! Print out keywords found
;    0511  5                        print_text (.linelevel, .level);    ! Print out text under key
;    0512  5                        rab [rab$b_rac] = rab$c_rfa ! Set random
;    0513  4                        END;
;    0514  4
;    0515  4                    nextdesc [dsc$a_pointer] = nextkey;     ! Set up lower key space
;    0516  5                    (
;    0517  5
;    0518  5                    LOCAL
;    0519  5                        stat;
;    0520  5
;    0521  5                    stat = do_help (.level + 1, rab [rab$w_rfa], nextdesc); ! Find lower level help
;    0522  4                    IF NOT .stat THEN RETURN .stat);
;    0523  4
;    0524  4                    IF .level EQL .helpinfo [hlp$l_realkeys]        ! If lowest level
;    0525  4                    THEN
;    0526  5                        (
;    0527  5
;    0528  5                        LOCAL
;    0529  5                            stat;
;    0530  5
;    0531  5                        stat = print_otherhelp (.level + 1, rab [rab$w_rfa], 1);    ! type lower keys
;    0532  4                        IF NOT .stat THEN RETURN .stat);
;    0533  4
;    0534  4                    IF NOT .wildkey AND .level LSS .helpinfo [hlp$l_allhelp] THEN RETURN 1;
;    0535  4
;    0536  4                    ! If this key wasn't wild, (even implicitly) don't continue searc' .ng
;    0537  4                    rab [rab$b_rac] = rab$c_rfa     ! Next loop read is rnd
;    0538  3                    END;                            ! Key found
;    0539  3
;    0540  2                END;                                ! Loop
;    0541  2
```

```
;    0542  2      !+
;    0543  2      ! Now, type out list of other keys at this level if:
;    0544  2      !       o There was no help found at this level (or a lower level)
;    0545  2      !         and this level is the highest wild-carded level or no levels
;    0546  2      !         are wildcarded.
;    0547  2      !       o If this is level 1 and the key was 'HELP' or if no text or
;    0548  2      !         other keys were printed out elsewhere.
;    0549  2      !-
;    0550  3          BEGIN
;    0551  3
;    0552  3          LOCAL
;    0553  3              position,
;    0554  3              size,
;    0555  3              first,
;    0556  3              wildpath;
;    0557  3
;    0558  3          BUILTIN
;    0559  3              FFS;
;    0560  3
;    0561  3          position = 0;
;    0562  3          size = .level - 1;
;    0563  3          wildpath = NOT FFS (position, size,          ! See if wild path here
;    0564  3              helpinfo [hlp$b_wildflags], first);
;    0565  3
;    0566  4          IF ( NOT .helpinfo [hlp$v_foundhelp]         ! If haven't found help
;    0567  5              AND NOT (.level GEQ .helpinfo [hlp$l_allhelp]   !  and not all-helped
;    0568  4              OR .wildpath))                           !  and not wild path
;    0569  3          THEN
;    0570  4              BEGIN                                    !                                  [03]
;    0571  4
;    0572  4              LOCAL
;    0573  4                  stat;
;    0574  4
;    0575  4              rab [rab$b_rac] = rab$c_rfa;             ! Set to random access             [03]
;    0576  4              CH$MOVE (6, saverfa, rab [rab$w_rfa]);   ! Copy begin RFA                   [03]
;    0577  4              stat = READ ();                          ! Position record stream           [03]
;    0578  4              stat = READ ();                          ! Advance past begin key           [03]
;    0579  4
;    0580  4              IF NOT .stat THEN RETURN .stat;          !                                  [03]
;    0581  4
;    0582  4              stat = print_otherhelp (.level, rab [rab$w_rfa], 0);    ! Type out other keys
;    0583  4
;    0584  4              IF NOT .stat THEN RETURN .stat
;    0585  4
;    0586  4              END                                      !                                  [03]
;    0587  2          END;
;    0588  2
;    0589  2          1
;    0590  1          END;                                         ! of do_help


                                                                  KEYNAMES=           HELPINFO+16

                                                 0FFC 00000 DO_HELP::.WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0352
                                    5E      FED8    CE   9E 00002            MOVAB   -296(SP), SP                        ;
```

```
                                5A 00000000'   EF   D0 00007          MOVL    HELPINFO+4, R10                          ; 0389
                                58         04  AC   D0 0000E          MOVL    LEVEL, R8                                ; 0390
                                5B 00000000'FF48   7E 00012          MOVAQ   @HELPINFO+36[R8], R11                    ;
                                5B             08  C2 0001A          SUBL2   #8, R11                                  ;
                                05             58  D1 0001D          CMPL    R8, #5                                   ; 0407
                                               03  15 00020          BLEQ    1$                                       ;
                                             01F2  31 00022          BRW     29$                                      ;
                      00000000'   EF           58  D1 00025 1$:      CMPL    R8, HELPINFO+40                          ;
                                               09  15 0002C          BLEQ    2$                                       ;
                      00000000'   EF           58  D1 0002E          CMPL    R8, HELPINFO                             ;
                                               78  19 00035          BLSS    12$                                      ;
                                         04    AB  D5 00037 2$:      TSTL    4(R11)                                   ; 0412
                                               06  13 0003A          BEQL    3$                                       ;
                                51       04    BB  90 0003C          MOVB    @4(R11), CHAR                            ; 0414
                                               02  11 00040          BRB     4$                                       ;
                                               51  94 00042 3$:      CLRB    CHAR                                     ; 0416
                                               50  D4 00044 4$:      CLRL    R0                                       ; 0418
                                2F             51  91 00046          CMPB    CHAR, #47                                ;
                                               02  12 00049          BNEQ    5$                                       ;
                                               50  D6 0004B          INCL    R0                                       ;
                                57             50  D0 0004D 5$:      MOVL    R0, QUALIFIER                            ;
                                56       FF    A8  9E 00050          MOVAB   -1(R8), LASTLEVEL                        ; 0419
                                6E             56  D0 00054          MOVL    LASTLEVEL, BOUND_LEVEL                   ;
                                   04    10    AA  9E 00057          MOVAB   16(R10), 4(SP)                           ; 0420
                04    BE         08    BC       06  28 0005C          MOVC3   #6, @FIRSTRFA, @4(SP)                    ;
                10    AE         08    AE       57  D2 00062          MCOML   QUALIFIER, 8(SP)                         ; 0438
                      AE         04    BE       06  28 00066 6$:      MOVC3   #6, @4(SP), SAVERFA                      ; 0428
                            0000V   CF           00  FB 0006C          CALLS   #0, READ                                 ; 0429
                                   0A            50  E8 00071          BLBS    STAT, 7$                                 ; 0431
                         0001827A   8F            50  D1 00074          CMPL    STAT, #98938                             ; 0434
                                                  23  13 0007B          BEQL    11$                                      ;
                                                      04 0007D          RET                                              ;
                                         18    AE   9F 0007E 7$:      PUSHAB  KEYDESC                                  ; 0436
                            0000V   CF           01  FB 00081          CALLS   #1, IS_KEY_ON_LINE                       ;
                                59               50  D0 00086          MOVL    R0, LINELEVEL                            ;
                                05         08    AE  E9 00089          BLBC    8(SP), 8$                                ; 0438
                                58               59  D1 0008D          CMPL    LINELEVEL, R8                            ;
                                                 08  11 00090          BRB     9$                                       ;
                                06               59  D1 00092 8$:      CMPL    LINELEVEL, #6                            ;
                                                 05  13 00095          BEQL    10$                                      ;
                                6E               59  D1 00097          CMPL    LINELEVEL, BOUND_LEVEL                   ; 0439
                                                 CA  14 0009A 9$:      BGTR    6$                                       ;
                                                 59  D5 0009C 10$:     TSTL    LINELEVEL                                ; 0440
                                                 C6  13 0009E          BEQL    6$                                       ;
                                58               59  D1 000A0 11$:     CMPL    LINELEVEL, R8                            ; 0442
                                                 0D  18 000A3          BGEQ    13$                                      ;
                                09         08    AE  E9 000A5          BLBC    8(SP), 13$                               ; 0443
                08    BC         04    BE       06  28 000A9          MOVC3   #6, @4(SP), @FIRSTRFA                    ; 0446
                                                0165  31 000AF 12$:     BRW     29$                                      ; 0447
                                   1E    AA       02  90 000B2 13$:     MOVB    #2, 30(R10)                              ; 0457
                                0000V   CF           00  FB 000B6 14$:     CALLS   #0, READ                                 ; 0467
                                               0D    50  E8 000BB          BLBS    STAT, 17$                                ; 0469
                         0001827A   8F            50  D1 000BE          CMPL    STAT, #98938                             ; 0472
                                                  03  12 000C5          BNEQ    16$                                      ;
                                                00FB  31 000C7 15$:     BRW     27$                                      ;
                                                      04 000CA 16$:     RET                                              ;
                                         18    AE   9F 000CB 17$:     PUSHAB  KEYDESC                                  ; 0474
```

```
                    0000V  CF              01  FB 000CE          CALLS   #1, IS_KEY_ON_LINE                                  ;
                           59              50  D0 000D3          MOVL    RC, LINELEVEL                                      ;
                           05              59  D1 000D6          CMPL    LINELEVEL, #5                                      ; 0476
                                           07  14 000D9          BGTR    18$                                                ;
                                           59  D5 000DB          TSTL    LINELEVEL                                          ; 0477
                                           03  15 000DD          BLEQ    18$                                                ;
                           56              59  D0 000DF          MOVL    LINELEVEL, LASTLEVEL                               ; 0479
                           6E              59  D1 000E2 18$:     CMPL    LINELEVEL, BOUND_LEVEL                             ; 0481
                                           04  14 000E5          BGTR    19$                                                ;
                                           59  D5 000E7          TSTL    LINELEVEL                                          ;
                                           DC  12 000E9          BNEQ    15$                                                ;
                           58              59  D1 000EB 19$:     CMPL    LINELEVEL, R8                                      ; 0483
                                           16  13 000EE          BEQL    20$                                                ;
                 00000000' EF              58  D1 000F0          CMPL    R8, HELPINFO                                       ; 0484
                                           0D  18 000F7          BGEQ    20$                                                ;
                           BA              57  E9 000F9          BLBC    QUALIFIER, 14$                                     ; 0485
                           06              59  D1 000FC          CMPL    LINELEVEL, #6                                      ; 0486
                                           B5  12 000FF          BNEQ    14$                                                ;
                           6E              56  D1 00101          CMPL    LASTLEVEL, BOUND_LEVEL                             ; 0487
                                           B0  12 00104          BNEQ    14$                                                ;
                              0C           AE  9F 00106 20$:     PUSHAB  WILDKEY                                            ; 0490
                                           5B  DD 00109          PUSHL   R11                                                ;
                              20           AE  9F 0010B          PUSHAB  KEYDESC                                            ;
                    0000V  CF              03  FB 0010E          CALLS   #3, DSR$KEY_EQUAL                                  ;
                           09              50  E8 00113          BLBS    R0, 21$                                            ;
                 00000000' EF              58  D1 00116          CMPL    R8, HELPINFO                                       ; 0491
                                           97  19 0011D          BLSS    14$                                                ;
                           50              0C  AC  D0 0011F 21$:     MOVL    KEY, R0                                        ; 0498
                           51                  6E  D0 00123          MOVL    BOUND_LEVEL, R1                                ;
                    00000000'EF41              50  D0 00126          MOVL    R0, KEYNAMES[R1]                               ;
                 51 00000000'  EF           1C  AE  C3 0012E          SUBL3   KEYDESC+4, HELPINFO+12, R1                     ; 0501
                                   51 00000000'  EF  C0 00137          ADDL2   HELPINFO+8, R1                                 ;
                           60                  51  B0 0013E          MOVW    R1, (R0)                                       ; 0500
                 04  B0        1C  BE          51  28 00141          MOVC3   R1, @KEYDESC+4, @4(R0)                         ; 0502
                                                50  D4 00147          CLRL    R0                                             ; 0503
                           06                  59  D1 00149          CMPL    LINELEVEL, #6                                  ;
                                               02  12 0014C          BNEQ    22$                                            ;
                                               50  D6 0014E          INCL    R0                                             ;
00000000'  EF              01                  50  F0 00150 22$:     INSV    R0, #1, #1, HELPINFO+45                        ;
                    00000000'  EF              58  D1 00159          CMPL    R8, HELPINFO+40                                ; 0505
                                               09  13 00160          BEQL    23$                                            ;
                    00000000'  FF              58  D1 00162          CMPL    R8, HELPINFO                                   ; 0506
                                               1B  19 00169          BLSS    24$                                            ;
                    00000000'  EF              04  88 0016B 23$:     BISB2   #4, HELPINFO+45                                ; 0509
                                               58  DD 00172          PUSHL   R8                                             ; 0510
                       0000V  CF               01  FB 00174          CALLS   #1, PRINT_KEYS                                 ;
                                               58  DD 00179          PUSHL   R8                                             ; 0511
                                               59  DD 0017B          PUSHL   LINELEVEL                                      ;
                       0000V  CF               02  FB 0017D          CALLS   #2, PRINT_TEXT                                 ;
                         1E   AA               02  90 00182          MOVB    #2, 30(R10)                                    ; 0512
                         24   AE           28  AE  9E 00186 24$:     MOVAB   NEXTKEY, NEXTDESC+4                            ; 0515
                                           20  AE  9F 0018B          PUSHAB  NEXTDESC                                       ; 0521
                                           08  AE  DD 0018E          PUSHL   8(SP)                                          ;
                                           01  A8  9F 00191          PUSHAB  1(R8)                                          ;
                       FE67   CF               03  FB 00194          CALLS   #3, DO_HELP                                    ;
                              7E               50  E9 00199          BLBC    STAT, 30$                                      ; 0522
                    00000000'  EF              58  D1 0019C          CMPL    R8, HELPINFO+40                                ; 0524
```

```
                                                    10  12 001A3          BNEQ    25$                                    ;
                                                    01  DD 001A5          PUSHL   #1                                     ; 0531
                                                08  AE  DD 001A7          PUSHL   8(SP)                                  ;
                                                01  A8  9F 001AA          PUSHAB  1(R8)                                  ;
                                        0000V   CF      03  FB 001AD          CALLS   #3, PRINT_OTHERHELP                    ;
                                                65      50  E9 001B2          BLBC    STAT, 30$                              ; 0532
                                                09      0C  AE  E8 001B5 25$:  BLBS    WILDKEY, 26$                           ; 0534
                                    00000000'   EF          58  D1 001B9          CMPL    R8, HELPINFO                           ;
                                                            55  19 001C0          BLSS    29$                            ;
                                                        FEED    31 001C2 26$:      BRW     13$                            ; 0537
                                                            52  D4 001C5 27$:      CLRL    POSITION                       ; 0561
                                                51          FF  A8  9E 001C7          MOVAB   -1(R8), SIZE                   ; 0562
                                                            50  D4 001CB          CLRL    R0                             ; 0564
                53 00000000'    EF              51          52  EA 001CD          FFS     POSITION, SIZE, HELPINFO+44, FIRST     ;
                                                            02  12 001D6          BNEQ    28$                            ;
                                                            50  D6 001D8          INCL    R0                             ;
                                                50          50  D2 001DA 28$:      MCOML   R0, WILDPATH                   ; 0563
                                    32 00000000'    EF      02  E0 001DD          BBS     #2, HELPINFO+45, 29$           ; 0566
                                       00000000'    EF      58  D1 001E5          CMPL    R8, HELPINFO                   ; 0567
                                                            29  18 001EC          BGEQ    29$                            ;
                                                26          50  E8 001EE          BLBS    WILDPATH, 29$                  ; 0568
                                                    1E      AA  02  90 001F1          MOVB    #2, 30(R10)                    ; 0575
                                            04  BE      10  AE  06  28 001F5          MOVC3   #6, SAVERFA, @4(SP)            ; 0576
                                                0000V   CF      00  FB 001FB          CALLS   #0, READ                       ; 0577
                                                0000V   CF      00  FB 00200          CALLS   #0, READ                       ; 0578
                                                12              50  E9 00205          BLBC    STAT, 30$                      ; 0580
                                                                7E  D4 00208          CLRL    -(SP)                          ; 0582
                                                            08  AE  DD 0020A          PUSHL   8(SP)                          ;
                                                            58  DD 0020D          PUSHL   R8                             ;
                                                0000V   CF      03  FB 0020F          CALLS   #3, PRINT_OTHERHELP            ;
                                                03              50  E9 00214          BLBC    STAT, 30$                      ; 0584
                                                50              01  D0 00217 29$:      MOVL    #1, R0                         ; 0590
                                                                    04 0021A 30$:      RET                                    ;

; Routine Size:  539 bytes,    Routine Base:  CODE + 025B
```

```
;    0591  1      %SBTTL 'check for key on line'
;    0592  1      ROUTINE is_key_on_line (key) =
;    0593  1
;    0594  1      !++
;    0595  1      ! FUNCTIONAL DESCRIPTION
;    0596  1      !       Scan the current line to see if it has a keyword on it.  The valid
;    0597  1      !       formats for a keyword line are "n key" and "/key" where n is a number
;    0598  1      !       from 1 to hlp$c_maxkeys, and key is a valid symbol (A-Z, a-z, $, _).
;    0599  1      !       It will return n, or hlp$c_maxkeys+1 if the qualifier form "/key" is
;    0600  1      !       found, or 0 if no key is found.
;    0601  1      !
;    0602  1      ! INPUTS
;    0603  1      !       key             address of descriptor of buffer for keyword
;    0604  1      !
;    0605  1      ! IMPLICIT INPUTS
;    0606  1      !       none
;    0607  1      !
;    0608  1      ! OUTPUTS
;    0609  1      !       none
;    0610  1      !
;    0611  1      ! IMPLICIT OUTPUTS
;    0612  1      !       none
;    0613  1      !
;    0614  1      ! SIDE EFFECTS
;    0615  1      !       none
;    0616  1      !
;    0617  1      ! RETURN CODES  (R0)
;    0618  1      !       level of key found
;    0619  1      !       hlp$c_maxkeys+1 if qualifier line
;    0620  1      !       0 if no key found
;    0621  1      !--
;    0622  1
```

```
;    0623  2          BEGIN
;    0624  2
;    0625  2          MAP
;    0626  2              key : REF bblock;
;    0627  2
;    0628  2          BIND
;    0629  2              line = helpinfo [hlp$l_bufdesc] : bblock;
;    0630  2
;    0631  2          LOCAL
;    0632  2              locdesc : bblock [dsc$c_s_bln],
;    0633  2              char : BYTE,
;    0634  2              level;
;    0635  2
;    0636  2          IF .line [dsc$w_length] EQL 0                ! If line is null,
;    0637  2          THEN
;    0638  2              RETURN 0;                                !  then can't be key on it
;    0639  2
;    0640  2          level = 0;                                   ! Initialize level
;    0641  2          CH$MOVE (dsc$c_s_bln, line, locdesc);        ! Initialize local descriptor
;    0642  2          char = CH$RCHAR (.locdesc [dsc$a_pointer]); ! Read first character
;    0643  2
;    0644  3          IF (.char LSSU %C'0'                         ! If not numeric
;    0645  2              OR .char GEQU %C'9') AND .char NEQ %C'/'          ! And not a qualifier line
;    0646  2          THEN
;    0647  2              RETURN 0                                 !  then it's not a key line
;    0648  2          ELSE
;    0649  3              BEGIN                                    !  else it IS a key line
;    0650  3
;    0651  3              IF .char NEQ %C'/'                       ! If it's a normal keyword
;    0652  3              THEN
;    0653  4                  BEGIN
;    0654  4
;    0655  4                  LOCAL
;    0656  4                      ptr,
;    0657  4                      len;
;    0658  4
;    0659  4                  ptr = .locdesc [dsc$a_pointer];      ! Get number ptr, len
;    0660  4                  len = scan_word (locdesc);           !  and scan past it
;    0661  4
;    0662  4                  WHILE .len GTR 0                     ! convert ascii to binary
;    0663  4                      AND CH$RCHAR (.ptr) GEQU %C'0' AND CH$RCHAR (.ptr) LEQU %C'9' DO
;    0664  5                      BEGIN
;    0665  5                      level = .level*10 + (CH$RCHAR_A (ptr) - %C'0');
;    0666  5                      len = .len - 1
;    0667  4                      END;
;    0668  4
;    0669  4                  IF .level GTR hlp$c_maxkeys          ! If level is illegally high,
;    0670  4                  THEN
;    0671  4                      RETURN 0;                        !  don't recognize it
;    0672  4
;    0673  4                  skip_blanks (locdesc)                ! Skip to keyword
;    0674  4                  END
;    0675  3              ELSE
;    0676  3                  level = hlp$c_maxkeys + 1;           ! Specify qualifier
;    0677  3
;    0678  3          key [dsc$a_pointer] = .locdesc [dsc$a_pointer]; ! Remember start of word
;    0679  3          key [dsc$w_length] = MIN (scan_word (locdesc), hlp$c_maxrecsiz) ! Length of string
```

```
;    0680  2          END;
;    0681  2
;    0682  2      .level
;    0683  1      END;                                    ! of is_key_on_line


                                                         LINE=                HELPINFO+8

                                                 00FC 00000 IS_KEY_ON_LINE:
                                                            .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0592
                          57 00000000'   EF  9E 00002       MOVAB   LINE, R7                              ;
                          5E             08  C2 00009       SUBL2   #8, SP                                ;
                                         67  B5 0000C       TSTW    LINE                                  ; 0636
                                         4D  13 0000E       BEQL    5$                                    ;
                                         56  D4 00010       CLRL    LEVEL                                 ; 0640
                       6E                67  08 28 00012    MOVC3   #8, LINE, LOCDESC                     ; 0641
                          50        04   BE  90 00016       MOVB    @LOCDESC+4, CHAR                      ; 0642
                          30             50  91 0001A       CMPB    CHAR, #48                             ; 0644
                                         05  1F 0001D       BLSSU   1$                                    ;
                          39             50  91 0001F       CMPB    CHAR, #57                             ; 0645
                                         05  1F 00022       BLSSU   2$                                    ;
                          2F             50  91 00024 1$:   CMPB    CHAR, #47                             ;
                                         34  12 00027       BNEQ    5$                                    ;
                          2F             50  91 00029 2$:   CMPB    CHAR, #47                             ; 0651
                                         3B  13 0002C       BEQL    7$                                    ;
                          52        04   AE  D0 0002E       MOVL    LOCDESC+4, PTR                        ; 0659
                                         5E  DD 00032       PUSHL   SP                                    ; 0660
                    0000V CF             01  FB 00034       CALLS   #1, SCAN_WORD                         ;
                          53             50  D0 00039       MOVL    R0, LEN                               ;
                                         1A  15 0003C 3$:   BLEQ    4$                                    ; 0662
                          30             62  91 0003E       CMPB    (PTR), #48                            ; 0663
                                         15  1F 00041       SSU     4$                                    ;
                          39             62  91 00043       CMPB    (PTR), #57                            ;
                                         10  1A 00046       BGTRU   4$                                    ;
                    50    56             0A  C5 00048       MULL3   #10, LEVEL, R0                        ; 0665
                          51             82  9A 0004C       MOVZBL  (PTR)+, R1                            ;
                          56        D0 A140  9E 0004F       MOVAB   -48(R1)[R0], LEVEL                    ;
                                         53  D7 00054       DECL    LEN                                   ; 0666
                                         E4  11 00056       BRB     3$                                    ;
                          05             56  D1 00058 4$:   CMPL    LEVEL, #5                             ; 0669
                                         03  15 0005B       BLEQ    6$                                    ;
                                         50  D4 0005D 5$:   CLRL    R0                                    ; 0671
                                             04 0005F       RET                                           ;
                                         5E  DD 00060 6$:   PUSHL   SP                                    ; 0673
                    0000V CF             01  FB 00062       CALLS   #1, SKIP_BLANKS                       ;
                                         03  11 00067       BRB     8$                                    ;
                          56             06  D0 00069 7$:   MOVL    #6, LEVEL                             ; 0676
                          52        04   AC  D0 0006C 8$:   MOVL    KEY, R2                               ; 0678
                       04 A2        04   AE  D0 00070       MOVL    LOCDESC+4, 4(R2)                      ;
                                         5E  DD 00075       PUSHL   SP                                    ; 0679
                    0000V CF             01  FB 00077       CALLS   #1, SCAN_WORD                         ;
                00000100 8F              50  D1 0007C       CMPL    R0, #256                              ;
                                         05  15 00083       BLEQ    9$                                    ;
                          50        0100 8F  3C 00085       MOVZWL  #256, R0                              ;
                          62             50  B0 0008A 9$:   MOVW    R0, (R2)                              ;
```

```
                                50          56  D0 0008D         MOVL    LEVEL, R0                                       ; 0683
                                                04 00090         RET                                                     ;

; Routine Size:  145 bytes,    Routine Base:  CODE + 0476
```

```
;    0684  1     %SBTTL 'print key list'
;    0685  1     ROUTINE print_keys (level) : NOVALUE =
;    0686  1
;    0687  1     !++
;    0688  1     ! FUNCTIONAL DESCRIPTION
;    0689  1     !       A list of pointers to string descriptors for keys along the path
;    0690  1     !       to the current level (i.e., key1 descriptor, key2 descriptor, etc.)
;    0691  1     !       is stored in the helpinfo block.  This routine will print, with
;    0692  1     !       the correct indentation, all keys along that path.
;    0693  1     !
;    0694  1     ! INPUTS
;    0695  1     !       level           The current level (lowest key name to output)
;    0696  1     !
;    0697  1     ! IMPLICIT INPUTS
;    0698  1     !       none
;    0699  1     !
;    0700  1     ! OUTPUTS
;    0701  1     !       none
;    0702  1     !
;    0703  1     ! IMPLICIT OUTPUTS
;    0704  1     !       none
;    0705  1     !
;    0706  1     ! SIDE EFFECTS
;    0707  1     !       none
;    0708  1     !
;    0709  1     ! RETURN CODES
;    0710  1     !       none
;    0711  1     !--
;    0712  1
```

```
;   0713  2          BEGIN
;   0714  2
;   0715  2          BIND
;   0716  2              keynames = helpinfo [hlp$l_keynames] : VECTOR [, LONG];
;   0717  2
;   0718  2          INCR i FROM 1 TO .level DO
;   0719  2
;   0720  2              !+
;   0721  2              ! Print; with trailing CRLF unless this is lowest level key and
;   0722  2              ! it's a qualifier.
;   0723  2              !-
;   0724  2
; P 0725  2              $ds_printf ((IF .helpinfo [hlp$v_qual] AND .i EQL .level THEN $ascic ('!/!#* !AS') ELSE $ascic (
;   0726  2                          '!/!#* !AS!/')), .i*2, .keynames [.i - 1]);
;   0727  2
;   0728  1          END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                        53  41  21  20  2A  23  21  2F  21  09  00015 P.AAE:  .ASCII  <9>\!/!#* !AS\                          ;
                2F  21  53  41  21  20  2A  23  21  2F  21  0B  0001F P.AAF:  .ASCII  <11>\!/!#* !AS!/\                        ;

                                                              KEYNAMES=         HELPINFO+16
                                                                .EXTRN  DS$PRINTF

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                0004 00000 PRINT_KEYS:
                                                                .WORD   Save R2                                         ; 0685
                                                     52  D4 00002          CLRL    I                                  ; 0718
                                                     32  11 00004          BRB     4$                                 ;
                                  00000000'EF        42  DD 00006 1$:      PUSHL   KEYNAMES-4[I]                      ; 0726
                        7E              52           01  78 0000D          ASHL    #1, I, -(SP)                       ;
                        0F 00000000'    EF           01  E1 00011          BBC     #1, HELPINFO+45, 2$                ;
                                  04    AC           52  D1 00019          CMPL    I, LEVEL                           ;
                                                     09  12 0001D          BNEQ    2$                                 ;
                                 50 00000000'        EF  9E 0001F          MOVAB   P.AAE, R0                          ;
                                                     07  11 00026          BRB     3$                                 ;
                                 50 00000000'        EF  9E 00028 2$:      MOVAB   P.AAF, R0                          ;
                                                     50  DD 0002F 3$:      PUSHL   R0                                 ;
                           00000000G  9F             03  FB 00031          CALLS   #3, @#DS$PRINTF                    ;
                        C9              52       04  AC  F3 00038 4$:      AOBLEQ  LEVEL, I, 1$                       ;
                                                     04  0003D            RET                                        ; 0728

; Routine Size:  62 bytes,    Routine Base:  CODE + 0507
```

```
;    0729  1      %SBTTL 'print help text'
;    0730  1      ROUTINE print_text (level, indent) =
;    0731  1
;    0732  1      !++
;    0733  1      ! FUNCTIONAL DESCRIPTION
;    0734  1      !       Print out help text.  Scan from given RFA until another key
;    0735  1      !       line is found, printing each line.  A qualifier key line which
;    0736  1      !       is imbedded within the text for a level is printed out as text.
;    0737  1      !
;    0738  1      ! INPUTS
;    0739  1      !       level           The level for which help is wanted (this can be
;    0740  1      !                       1 to hlp$c_maxkeys; or hlp$c_maxkeys+1 for
;    0741  1      !                       qualifier help)
;    0742  1      !       indent          position of key in input list
;    0743  1      !
;    0744  1      ! IMPLICIT INPUTS
;    0745  1      !       none
;    0746  1      !
;    0747  1      ! OUTPUTS
;    0748  1      !
;    0749  1      ! IMPLICIT UUTPUTS
;    0750  1      !       none
;    0751  1      !
;    0752  1      ! SIDE EFFECTS
;    0753  1      !       none
;    0754  1      !
;    0755  1      ! RETURN CODES
;    0756  1      !       none
;    0757  1      !--
;    0758  1
```

```
;    0759  2          BEGIN
;    0760  2
;    0761  2          LOCAL
;    0762  2              stat,
;    0763  2              logtabs,                                ! Number of logical tabs to indent
;    0764  2              linelevel,                              ! Level of key record
;    0765  2              lastqual,                               ! Flag for qualifier help text
;    0766  2              keydesc : bblock [dsc$c_s_bln];         ! Descriptor for key
;    0767  2
;    0768  2          logtabs = .indent + (IF .level LEQ hlp$c_maxkeys THEN 1 ELSE 0);
;    0769  2
;    0770  2          !+
;    0771  2          ! Now print out lines until we find another qualifier.  Note that
;    0772  2          ! unless we are looking for qualifier help, or we are in ALLHELP
;    0773  2          ! mode, a qualifier key line counts as text!
;    0774  2          !-
;    0775  2
;    0776  2          lastqual = 1;                               ! Pre-set switch so it works
;    0777  2
;    0778  2          WHILE
;    0779  3              BEGIN
;    0780  3              stat = READ ();                         ! Read next record
;    0781  3
;    0782  3              IF NOT .stat AND .stat NEQ rms$_eof THEN RETURN .stat;
;    0783  3
;    0784  4              BEGIN
;    0785  4
;    0786  4              IF NOT .stat
;    0787  4              THEN
;    0788  4                  0                                   ! If end of file, no more text
;    0789  4              ELSE
;    0790  5                  BEGIN
;    0791  5                  linelevel = is_key_on_line (keydesc);       ! Get help key
;    0792  6                  (.linelevel EQL 0                   ! continue if no key on line
;    0793  7                  OR (.level EQL hlp$c_maxkeys + 1 AND .lastqual)     ! or second qual in row
;    0794  7                  OR (.linelevel EQL hlp$c_maxkeys + 1        ! or qual line and not printing
;    0795  6                  AND .level LEQ hlp$c_maxkeys))      ! qualifier help
;    0796  5                  END
;    0797  5
;    0798  4              END
;    0799  3              END
;    0800  2          DO
;    0801  3              BEGIN
;    0802  3              lastqual = (IF .linelevel EQL hlp$c_maxkeys + 1 THEN 1 ELSE 0);
;    0803  4              $ds_printf ($ascic ('!/!#* !AS'), .logtabs*2, helpinfo [hlp$l_bufdesc])
;    0804  2              END;
;    0805  2
;    0806  2          print_crlf ();                              ! Force double space after text
;    0807  2          .stat
;    0808  1          END;                                        ! of print_text


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                 53  41  21  20  2A  23  21  2F  21  09  0002B P.AAG:  .ASCII  <9>\!/!#* !AS\                          ;
```

```
                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                OFFC 00000 PRINT_TEXT:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 0730
                                    5E          08  C2 00002          SUBL2   #8, SP                          ;
                                                56  D4 00005          CLRL    R6                              ; 0768
                                    05      04  AC  D1 0C007          CMPL    LEVEL, #5                       ;
                                                07  14 0000B          BGTR    1$                              ;
                                                56  D6 0000D          INCL    R6                              ;
                                    50          01  D0 0000F          MOVL    #1, R0                          ;
                                                02  11 00012          BRB     2$                              ;
                                                50  D4 00014 1$:      CLRL    R0                              ;
                                    50      08  AC  C0 00016 2$:      ADDL2   INDENT, LOGTABS                 ;
                                    52          01  D0 0001A          MOVL    #1, LASTQUAL                    ; 0776
                                55              50  01  78 0001D      ASHL    #1, LOGTABS, R5                 ; 0803
                                      0000V     CF  00  FB 00021 3$:  CALLS   #0, READ                        ; 0780
                                    53          50  D0 00026          MOVL    R0, STAT                        ;
                                    0C          53  E8 00029          BLBS    STAT, 4$                        ; 0782
                                 0001827A       8F  53  D1 0002C      CMPL    STAT, #98938                    ;
                                                46  12 00033          BNEQ    10$                             ;
                                    40          53  E9 00035          BLBC    STAT, 9$                        ; 0786
                                    5E          DD  00038 4$:         PUSHL   SP                              ; 0791
                                    FEF2        CF  01  FB 0003A      CALLS   #1, IS_KEY_ON_LINE              ;
                                    54          50  D0 0003F          MOVL    R0, LINELEVEL                   ;
                                                11  13 00042          BEQL    6$                              ; 0792
                                    06      04  AC  D1 00044          CMPL    LEVEL, #6                       ; 0793
                                                03  12 00048          BNEQ    5$                              ;
                                    08          52  E8 0004A          BLBS    LASTQUAL, 6$                    ;
                                    06          54  D1 0004D 5$:      CMPL    LINELEVEL, #6                   ; 0794
                                                26  12 00050          BNEQ    9$                              ;
                                    23          56  E9 00052          BLBC    R6, 9$                          ; 0795
                                    06          54  D1 00055 6$:      CMPL    LINELEVEL, #6                   ; 0802
                                                05  12 00058          BNEQ    7$                              ;
                                    52          01  D0 0005A          MOVL    #1, LASTQUAL                    ;
                                                02  11 0005D          BRB     8$                              ;
                                                52  D4 0005F 7$:      CLRL    LASTQUAL                        ;
                                    00000000'   EF  9F 00061 8$:      PUSHAB  HELPINFO+8                      ; 0803
                                                55  DD 00067          PUSHL   R5                              ;
                                    00000000'   EF  9F 00069          PUSHAB  P.AAG                           ;
                          00000000G 9F              03  FB 0006F      CALLS   #3, @#DS$PRINTF                 ;
                                                A9  11 00076          BRB     3$                              ;
                                              0000V 30  00078 9$:     BSBW    PRINT_CRLF                      ; 0806
                                    50            53  D0 0007B 10$:    MOVL    STAT, R0                        ; 0808
                                                  04  0007E           RET                                     ;

; Routine Size:  127 bytes,    Routine Base:  CODE + 0545
```

```
;    0809  1     %SBTTL 'List other keys'
;    0810  1     ROUTINE print_otherhelp (level, keyrfa, errflg) =
;    0811  1
;    0812  1     !++
;    0813  1     ! FUNCTIONAL DESCRIPTION
;    0814  1     !       When the search for a help key fails, or if the key is level 1, and
;    0815  1     !       is 'HELP', a list is printed of all 'other' help keys valid for
;    0816  1     !       that level.
;    0817  1     !
;    0818  1     ! INPUTS
;    0819  1     !       level           The level for which keys are to be listed
;    0820  1     !       keyrfa          The RFA to begin scan at (immediately following the
;    0821  1     !                       RFA for the next highest level)
;    0822  1     !       errflg          0 if 'no keys found' message should be typed
;    0823  1     !
;    0824  1     ! IMPLICIT INPUTS
;    0825  1     !       none
;    0826  1     !
;    0827  1     ! OUTPUTS
;    0828  1     !
;    0829  1     ! IMPLICIT OUTPUTS
;    0830  1     !       none
;    0831  1     !
;    0832  1     ! SIDE EFFECTS
;    0833  1     !       Updates RAB RFA past keys
;    0834  1     !
;    0835  1     ! RETURN CODES
;    0836  1     !       none
;    0837  1     !--
;    0838  1
```

```
;    0839  2          BEGIN
;    0840  2
;    0841  2          BIND
;    0842  2              rab = .helpinfo [hlp$l_rab] : bblock,
;    0843  2              fmt = $ascic ('!/  !AD'),                   ! Format output line
;    0844  2              otherinfo = $ascic ('!/  Additional information available:!/');
;    0845  2
;    0846  2          LOCAL
;    0847  2              qualine,                                    ! Flag for qualifier line
;    0848  2              no_qual_yet,                                ! Flag for qual. line found
;    0849  2              lastlevel,                                  ! Last key level found
;    0850  2              linelevel,                                  ! Level of current record
;    0851  2              first,                                      ! First-time flag
;    0852  2              linecol,                                    ! Current column of line
;    0853  2              ptr,                                        ! Pointer to output buffer
;    0854  2              keydesc : bblock [dsc$c_s_bln],             ! Key descriptor
;    0855  2              outbuf : VECTOR [hlp$c_maxrecsiz, BYTE];        ! Output buffer
;    0856  2
```

```
;    0857  2     %SBTTL 'PrintHeader'
;    0858  2         ROUTINE printheader (err, lev) : jsb_2 NOVALUE =
;    0859  3             BEGIN
;    0860  3
;    0861  3             BIND
;    0862  3                 nodocmsg = $ascic ('!/  Sorry, no documentation on !AS !AS !AS !AS !AS!/'),
;    0863  3                 keylist = .helpinfo [hlp$l_keylist] : VECTOR [, LONG];
;    0864  3
;    0865  3             IF NOT .err
;    0866  3             THEN
;    0867  4                 BEGIN
;    0868  4                 helpinfo [hlp$v_foundhelp] = 1;      ! Say we found something.
;    0869  4                 print_keys (.lev - 1);               ! Print keys to next higher
;    0870  5                 $ds_printf (nodocmsg, keylist [0], keylist [2], keylist [4], keylist [6], keylist [8])
;    0871  5                                                      ! Print out "no help" msg
;    0872  3                 END;
;    0873  3
;    0874  4             $ds_printf (otherinfo)
;    0875  2             END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                    44 41 21 20 20 2F 21 07  00035 P.AAH:   .ASCII  <7>\!/  !AD\                    ;
6C 61 6E 6F 69 74 69 64 64 41 20 20 2F 21 27  0003D P.AAI:   .ASCII  \'!/  Additional information available:!/\   ;
76 61 20 6E 6F 69 74 61 6D 72 6F 66 6E 69 20  0004C                                                              ;
                 2F 21 3A 65 6C 62 61 6C 69 61  0005B                                                            ;
20 6F 6E 20 2C 79 72 72 6F 53 20 20 2F 21 34  00065 P.AAJ:   .ASCII  \4!/  Sorry, no documentation on !AS !AS \   ;
6F 20 6E 6F 69 74 61 74 6E 65 6D 75 63 6F 64  00074                                                              ;
                 20 53 41 21 20 53 41 21 20 6E  00083                                                            ;
       2F 21 53 41 21 20 53 41 21 20 53 41 21  0008D            .ASCII  \!AS !AS !AS!/\                         ;

                                                        FMT=                    P.AAH
                                                        OTHERINFO=              P.AAI
                                                        NODOCMSG=               P.AAJ


                                                                .PSECT  CODE,NOWRT,  SHR,2

                                        52  DD  00000  PRINTHEADER:
                                                        PUSHL   R2                                              ; 0858
                                52 00000000'    EF  D0  00002           MOVL    HELPINFO+36, R2                 ; 0863
                                            2A  50  E8  00009           BLBS    ERR, 1$                         ; 0865
                    00000000'   EF              04  88  0000C           BISB2   #4, HELPINFO+45                 ; 0868
                                        FF      A1  9F  00013           PUSHAB  -1(LEV)                         ; 0869
                        FF28    CF              01  FB  00016           CALLS   #1, PRINT_KEYS                  ;
                                        20      A2  9F  0001B           PUSHAB  32(R2)                          ; 0870
                                        18      A2  9F  0001E           PUSHAB  24(R2)                          ;
                                        10      A2  9F  00021           PUSHAB  16(R2)                          ;
                                        08      A2  9F  00024           PUSHAB  8(R2)                           ;
                                        52      DD  00027           PUSHL   R2                              ;
                                00000000'       EF  9F  00029           PUSHAB  NODOCMSG                        ;
                    00000000G   9F              06  FB  0002F           CALLS   #6, @#DS$PRINTF                 ;
                                00000000'       EF  9F  00036 1$:       PUSHAB  OTHERINFO                       ; 0874
                    00000000G   9F              01  FB  0003C           CALLS   #1, @#DS$PRINTF                 ;
                                            04  BA  00043           POPR    #^M<R2>                         ; 0875
```

```
                                        05 00045        RSB                                             ;

; Routine Size:  70 bytes,    Routine Base:  CODE + 05C4
```

```
;    0876  2      %SBTTL 'print_otherhelp main code'
;    0877  2
;    0878  2          IF .level GEQ .helpinfo [hlp$l_allhelp] OR .helpinfo [hlp$v_qual] THEN RETURN 1;
;    0879  2
;    0880  2          first = 1;                                      ! Set first-time flag
;    0881  2          lastlevel = .level - 1;                         ! Init last key found
;    0882  2          no_qual_yet = 1;                                ! Haven't found qualifier yet
;    0883  2          ptr = CH$PTR (outbuf);
;    0884  2          linecol = 0;                                    ! Set column pointer
;    0885  2          CH$MOVE (rfa$c_length, .keyrfa, rab [rab$w_rfa]);   ! Initialize local RFA
;    0886  2          rab [rab$b_rac] = rab$c_rfa;                    ! Set random access
;    0887  2
;    0888  2          WHILE
;    0889  3              BEGIN
;    0890  3
;    0891  3              LOCAL
;    0892  3                  stat;
;    0893  3
;    0894  3              stat = READ ();                             ! Read next record
;    0895  3              rab [rab$b_rac] = rab$c_seq;                ! Set back to sequential
;    0896  3
;    0897  3              IF NOT .stat AND .stat NEQ rms$_eof THEN RETURN .stat;
;    0898  3
;    0899  4              BEGIN
;    0900  4
;    0901  4              IF NOT .stat
;    0902  4              THEN
;    0903  5                  (0)
;    0904  4              ELSE
;    0905  5                  BEGIN
;    0906  5                  linelevel = is_key_on_line (keydesc);
;    0907  5
;    0908  5                  IF (.linelevel GTR 0) AND (.linelevel LEQ hlp$c_maxkeys) THEN lastlevel = .linelevel;
;    0909  5
;    0910  5                  qualine = (IF (.linelevel LEQ hlp$c_maxkeys) OR (.lastlevel NEQ .level) THEN 0 ELSE 1);
;    0911  6                  ((.linelevel GEQ .level) OR (.linelevel EQL 0))
;    0912  5                  END
;    0913  5
;    0914  4              END
;    0915  3              END
;    0916  2          DO
;    0917  2
;    0918  2              IF (.linelevel EQL .level) OR .qualine  ! Type keys & qualifiers
;    0919  2              THEN
;    0920  3                  BEGIN
;    0921  3
;    0922  3                  LOCAL
;    0923  3                      length,
;    0924  3                      reallen,
;    0925  3                      outlen;
;    0926  3
;    0927  3                  reallen = .helpinfo [hlp$l_bufdesc] - (.keydesc [dsc$a_pointer] - .(helpinfo [hlp$l_bufdesc] + 4));
;    0928  3                                                          ! Length of rest of record
;    0929  3                  !+
;    0930  3                  ! At this point, the following are true:
;    0931  3                  !   RealLen = the length of the key word or the qualifier to be printed
;    0932  3                  !   LineCol = Current column in the line (starting at 0)
```

```
;    0933  3                    !-
;    0934  3
;    0935  3                    !+
;    0936  3                    ! Now set Length to the length of the key word plus at least one space
;    0937  3                    ! and make the Length modulo 15. That is, if the RealLen of the key word
;    0938  3                    ! is 8, the Length will be 15 (15 mod 15 = 0). If the RealLen of the key
;    0939  3                    ! word is 15, the Length will be 30 (30 mod 15 = 0). In this second
;    0940  3                    ! example, the one space required after the key word makes it 16 plus the
;    0941  3                    ! additional spaces (for which 14 wuld be needed to bring it to the next
;    0942  3                    ! field).
;    0943  3                    !-
;    0944  3
;    0945  3                    length = ((.reallen + 15) / 15) * 15;
;    0946  3
;    0947  3                    outlen = CH$DIFF (.ptr, CH$PTR (outbuf));
;    0948  3
;    0949  4                    IF ((.outlen + .length) GTR 78) OR (.no_qual_yet AND .qualine)
;    0950  3                    THEN
;    0951  4                        BEGIN
;    0952  4
;    0953  4                        IF .qualine THEN no_qual_yet = 0;
;    0954  4
;    0955  4                        IF .first THEN printheader (.errflg, .level);
;    0956  4
;    0957  4                        first = 0;
;    0958  4                        $ds_printf (fmt, .outlen, outbuf);
;    0959  4                        ptr = CH$PTR (outbuf);
;    0960  4                        linecol = 0
;    0961  3                        END;
;    0962  3
;    0963  3                    ptr = CH$MOVE (.reallen, .keydesc [dsc$a_pointer], %C' ', .length, .ptr);
;    0964  3                                                    ! Copy key into output descriptor
;    0965  3                    linecol = .linecol + .length    ! Update column count
;    0966  2                    END;
;    0967  2
;    0968  2                IF .linecol GTR 0                   ! If we have a partial line left,
;    0969  2                THEN
;    0970  3                    BEGIN
;    0971  3
;    0972  3                    IF .first THEN printheader (.errflg, .level);
;    0973  3
;    0974  4                    $ds_printf ($ascic ('!/  !AD!/'), CH$DIFF (.ptr, CH$PTR (outbuf)), outbuf)
;    0975  2                    END;
;    0976  2
;    0977  2                1
;    0978  1                END;                                ! of print_otherhelp


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2
                2F  21  44  41  21  20  20  2F  21  09  0009A P.AAK:  .ASCII  <9>\!/  !AD!/\                          ;

                                                                .PSECT  CODE,NOWRT,  SHR,2
```

```
                                   0FFC 00000 PRINT_OTHERHELP:
                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11       ; 0810
                 5E     FEE8   CE   9E 00002          MOVAB   -280(SP), SP                               ;
                 56 00000000'  EF   D0 00007          MOVL    HELPINFO+4, R6                             ; 0842
                 57       04   AC   D0 0000E          MOVL    LEVEL, R7                                  ; 0878
         00000000'  EF             57   D1 00012          CMPL    R7, HELPINFO                               ;
                    03             19 00019          BLSS    2$                                         ;
                             011A   31 0001B 1$:      BRW     18$                                        ;
      F5 00000000'  EF             01   E0 0001E 2$:      BBS     #1, HELPINFO+45, 1$                        ;
                 04  AE             01   D0 00026          MOVL    #1, FIRST                                  ; 0880
                 08  AE       FF   A7   9E 0002A          MOVAB   -1(R7), LASTLEVEL                          ; 0881
                 0C  AE             01   D0 0002F          MOVL    #1, NO_QUAL_YET                            ; 0882
                     5A       10   AE   9E 00033          MOVAB   OUTBUF, PTR                                ; 0883
                                   6E   D4 00037          CLRL    LINECOL                                    ; 0884
   10      A6        08  BC             06   28 00039          MOVC3   #6, @KEYRFA, 16(R6)                        ; 0885
                     1E  A6             02   90 0003F          MOVB    #2, 30(R6)                                 ; 0886
               0000V  CF             00   FB 00043 3$:      CALLS   #0, READ                                   ; 0894
                            1E   A6   94 00048          CLRB    30(R6)                                     ; 0895
                     10             50   E8 0004B          BLBS    STAT, 5$                                   ; 0897
         0001827A   8F             50   D1 0004E          CMPL    STAT, #98938                               ;
                     01             13 00055          BEQL    4$                                         ;
                                   04 00057          RET                                                ;
                     03             50   E8 00058 4$:      BLBS    STAT, 5$                                   ; 0901
                             00B0   31 0005B          BRW     16$                                        ;
                            F8   AD   9F 0005E 5$:      PUSHAB  KEYDESC                                    ; 0906
                FE06   CF             01   FB 00061          CALLS   #1, IS_KEY_ON_LINE                         ;
                     59             50   D0 00066          MOVL    R0, LINELEVEL                              ;
                     09             15 00069          BLEQ    6$                                         ; 0908
                     05             59   D1 0006B          CMPL    LINELEVEL, #5                              ;
                     04             14 0006E          BGTR    6$                                         ;
                 08  AE             59   D0 00070          MOVL    LINELEVEL, LASTLEVEL                       ;
                     05             59   D1 00074 6$:      CMPL    LINELEVEL, #5                              ; 0910
                     06             15 00077          BLEQ    7$                                         ;
                     57       08   AE   D1 00079          CMPL    LASTLEVEL, R7                              ;
                     04             13 0007D          BEQL    8$                                         ;
                                   5B   D4 0007F 7$:      CLRL    QUALINE                                    ;
                     03             11 00081          BRB     9$                                         ;
                     5B             01   D0 00083 8$:      MOVL    #1, QUALINE                                ;
                     57             59   D1 00086 9$:      CMPL    LINELEVEL, R7                              ; 0911
                     04             18 00089          BGEQ    10$                                        ;
                                   59   D5 0008B          TSTL    LINELEVEL                                  ;
                     7F             12 0008D          BNEQ    16$                                        ;
                     57             59   D1 0008F 10$:     CMPL    LINELEVEL, R7                              ; 0918
                     03             13 00092          BEQL    11$                                        ;
                     AC             5B   E9 00094          BLBC    QUALINE, 3$                                ;
      50 00000000'  EF       FC   AD   C3 00097 11$:     SUBL3   KEYDESC+4, HELPINFO+12, R0                 ; 0927
      53              50 00000000'  EF   C1 000A0          ADDL3   HELPINFO+8, R0, REALLEN                    ;
                     50       0F   A3   9E 000A8          MOVAB   15(R3), R0                                 ; 0945
                     50             0F   C6 000AC          DIVL2   #15, R0                                    ;
      58             50             0F   C5 000AF          MULL3   #15, R0, LENGTH                            ;
                     50       10   AE   9E 000B3          MOVAB   OUTBUF, R0                                 ; 0947
      52             5A             50   C3 000B7          SUBL3   R0, PTR, OUTLEN                            ;
      50             52             58   C1 000BB          ADDL3   LENGTH, OUTLEN, R0                         ; 0949
         0000004E   8F             50   D1 000BF          CMPL    R0, #78                                    ;
                     07             14 000C6          BGTR    12$                                        ;
                     32       0C   AE   E9 000C8          BLBC    NO_QUAL_YET, 15$                           ;
                     2F             5B   E9 000CC          BLBC    QUALINE, 15$                               ;
```

```
                                03              5B  E9 000CF 12$:    BLBC    QUALINE, 13$                                    ; 0953
                                            0C  AE  D4 000D2         CLRL    NO_QUAL_YET                                     ;
                                0A          04  AE  E9 000D5 13$:    BLBC    FIRST, 14$                                      ; 0955
                                51              57  D0 000D9         MOVL    R7, R1                                          ;
                                50          0C  AC  D0 000DC         MOVL    ERRFLG, R0                                      ;
                                                FED7  30 000E0         BSBW    PRINTHEADER                                     ;
                                            04  AE  D4 000E3 14$:    CLRL    FIRST                                           ; 0957
                                            10  AE  9F 000E6         PUSHAB  OUTBUF                                          ; 0958
                                                52  DD 000E9         PUSHL   OUTLEN                                          ;
                                00000000'       EF  9F 000EB         PUSHAB  FMT                                             ;
                        00000000G   9F          03  FB 000F1         CALLS   #3, @#DS$PRINTF                                 ;
                                5A          10  AE  9E 000F8         MOVAB   OUTBUF, PTR                                     ; 0959
                                            6E  D4 000FC             CLRL    LINECOL                                         ; 0960
        58          20      FC  BD              53  2C 000FE 15$:    MOVC5   REALLEN, @KEYDESC+4, #32, LENGTH, (PTR)         ; 0963
                                                6A      00104        ;                                                       ;
                                5A              53  D0 00105         MOV     R3, PTR                                         ;
                                6E              58  C0 00108         ADDL2   LENGTH, LINECOL                                 ; 0965
                                                FF35  31 0010B         BRW     3$                                              ; 0918
                                                6E  D5 0010E 16$:    TSTL    LINECOL                                         ; 0968
                                                26  15 00110         BLEQ    18$                                             ;
                                0A          04  AE  E9 00112         BLBC    FIRST, 17$                                      ; 0972
                                51              57  D0 00116         MOVL    R7, R1                                          ;
                                50          0C  AC  D0 00119         MOVL    ERRFLG, R0                                      ;
                                                FE9A  30 0011D         BSBW    PRINTHEADER                                     ;
                                            10  AE  9F 00120 17$:    PUSHAB  OUTBUF                                          ; 0974
                                50          14  AE  9E 00123         MOVAB   OUTBUF, R0                                      ;
                        7E              5A          50  C3 00127         SUBL3   R0, PTR, -(SP)                              ;
                                00000000'       EF  9F 0012B         PUSHAB  P.AAK                                           ;
                        00000000G   9F          03  FB 00131         CALLS   #3, @#DS$PRINTF                                 ;
                                50              01  D0 00138 18$:    MOVL    #1, R0                                          ; 0978
                                                    04 0013B         RET                                                     ;

; Routine Size:  316 bytes,    Routine Base:  CODE + 060A
```

```
;    0979  1     %SBTTL 'Read random record'
;    0980  1     ROUTINE READ =
;    0981  1
;    0982  1     !++
;    0983  1     ! FUNCTIONAL DESCRIPTION
;    0984  1     !       Read a logical record from the current help file
;    0985  1     !
;    0986  1     ! INPUTS
;    0987  1     !
;    0988  1     ! IMPLICIT INPUTS
;    0989  1     !       helpinfo        HELP module storage block
;    0990  1     !       RAB
;    0991  1     !
;    0992  1     ! OUTPUTS
;    0993  1     !       none
;    0994  1     !
;    0995  1     ! IMPLICIT OUTPUTS
;    0996  1     !       helpinfo [hlp$l_bufdesc] is set to describe the record read.
;    0997  1     !
;    0998  1     ! SIDE EFFECTS
;    0999  1     !       none
;    1000  1     !
;    1001  1     ! RETURN CODES  (R0)
;    1002  1     !       SS$_NORMAL
;    1003  1     !       any RMS error code
;    1004  1     !--
;    1005  1
```

```
;    1006  2          BEGIN
;    1007  2
;    1008  2          LOCAL
;    1009  2              len,
;    1010  2              ptr,
;    1011  2              stat;
;    1012  2
;    1013  2          BIND
;    1014  2              rab = .helpinfo [hlp$l_rab] : bblock,    ! Map to RAB
;    1015  2              fab = .rab [rab$l_fab] : bblock;         ! Map to FAB
;    1016  2
;    1017  2          stat = $get (rab = rab);                     ! Read a record
;    1018  2          rab [rab$b_rac] = rab$c_seq;                 ! Restore sequential access
;    1019  2          len = .rab [rab$w_rsz];                      ! Get length
;    1020  2          ptr = .rab [rab$l_rbf];                      !  and address
;    1021  2
;    1022  2          IF .fab [fab$b_rat] EQL fab$m_ftn            ! If fortran format
;    1023  2              AND .len GTR 0                           !  and not null record
;    1024  2          THEN
;    1025  3              BEGIN
;    1026  3              len = .len - 1;                          ! Lose the CR byte
;    1027  3              ptr = .ptr + 1
;    1028  2              END;
;    1029  2
;    1030  2          IF .fab [fab$b_rat] EQL 0                    ! If no attribute
;    1031  2              AND .len GEQ 2                           !.. and there are two chars                              [06]
;    1032  2              AND .(.ptr + .len - 2)<0, 16> EQL %X'0a0d'       ! .. and last chars are CRLF
;    1033  2          THEN
;    1034  2              len = .len - 2;                          ! Slice off CRLF
;    1035  2
;    1036  2          helpinfo [hlp$l_bufdesc] = .len;             ! Length read
;    1037  2          helpinfo [hlp$l_bufdesc] + 4 = .ptr;
;    1038  2
;    1039  2          IF .stat EQL rms$_rfa THEN stat = rms$_eof; ! Fake error type if bad RFA
;    1040  2
;    1041  2          .stat
;    1042  1          END;                                         ! of read


                                                                .EXTRN  SYS$GET

                                                  000C 00000 READ:  .WORD   Save R2,R3                                ;  0980
                                  52 00000000'  EF   D0 00002       MOVL    HELPINFO+4, R2                            ;  1014
                                  53        3C  A2   D0 00009       MOVL    60(R2), R3                                ;  1015
                                              52   DD 0000D       PUSHL   R2                                        ;  1017
                        00000000G 00          01   FB 0000F       CALLS   #1, SYS$GET                               ;
                                          1E  A2   94 00016       CLRB    30(R2)                                    ;  1018
                                  51      22  A2   3C 00019       MOVZWL  34(R2), LEN                               ;  1019
                                  52      28  A2   D0 0001D       MOVL    40(R2), PTR                               ;  1020
                                  01      1E  A3   91 00021       CMPB    30(R3), #1                                ;  1022
                                              08   12 00025       BNEQ    1$                                        ;
                                              51   D5 00027       TSTL    LEN                                       ;  1023
                                              04   15 00029       BLEQ    1$                                        ;
                                              51   D7 0002B       DECL    LEN                                       ;  1026
                                              52   D6 0002D       INCL    PTR                                       ;  1027
                                          1E  A3   95 0002F 1$:     TSTB    30(R3)                                    ;  1030
```

```
                                        13  12 00032          BNEQ    2$                                      ;
                                02      51  D1 00034          CMPL    LEN, #2                                 ; 1031
                                        0E  19 00037          BLSS    2$                                      ;
                                    FE A142 9F 00039          PUSHAB  -2(LEN)[PTR]                            ; 1032
                            0A0D    8F      9E B1 0003D       CMPW    @(SP)+, #2573                           ;
                                            03  12 00042      BNEQ    2$                                      ;
                                51          02  C2 00044      SUBL2   #2, LEN                                 ; 1034
                    00000000'   EF          51  7D 00047 2$:  MOVQ    LEN, HELPINFO+8                         ; 1036
                    0001865C    8F          50  D1 0004E      CMPL    STAT, #99932                            ; 1039
                                            07  12 00055      BNEQ    3$                                      ;
                                50 0001827A 8F  D0 00057      MOVL    #98938, STAT                            ;
                                            04  0005E 3$:     RET                                             ; 1042

; Routine Size:  95 bytes,    Routine Base:  CODE + 0746
```

```
;    1043  1     %SBTTL 'compare keys'
;    1044  1
;    1045  1     GLOBAL ROUTINE dsr$key_equal (key1, key2, wildcompare) =
;    1046  1
;    1047  1     !++
;    1048  1     ! FUNCTIONAL DESCRIPTION
;    1049  1     !       Using wildcard characters ('*' and '%') compare two strings.
;    1050  1     !
;    1051  1     ! INPUTS
;    1052  1     !       key1            The non-wildcard match string
;    1053  1     !       key2            The pattern string (possibly wildcarded)
;    1054  1     !       wildcompare     Address of flag to set if wildcarded compare
;    1055  1     !
;    1056  1     ! IMPLICIT INPUTS
;    1057  1     !       none
;    1058  1     !
;    1059  1     ! OUTPUTS
;    1060  1     !       none
;    1061  1     !
;    1062  1     ! IMPLICIT OUTPUTS
;    1063  1     !       none
;    1064  1     !
;    1065  1     ! SIDE EFFECTS
;    1066  1     !       none
;    1067  1     !
;    1068  1     ! RETURN CODES  (R0)
;    1069  1     !       0       strings do not match
;    1070  1     !       1       strings match
;    1071  1     !--
;    1072  1
```

```
;    1073  2          BEGIN
;    1074  2
;    1075  2          MAP
;    1076  2              key1 : REF bblock,
;    1077  2              key2 : REF bblock;
;    1078  2
;    1079  2          LABEL
;    1080  2              loop;
;    1081  2
;    1082  2          LOCAL
;    1083  2              wild,
;    1084  2              len1,
;    1085  2              len2,
;    1086  2              ptr1,
;    1087  2              ptr2,
;    1088  2              savlen1,
;    1089  2              savlen2,
;    1090  2              savptr1,
;    1091  2              savptr2;
;    1092  2
;    1093  2          .wildcompare = wild = 0;                    ! Init wild flag
;    1094  2          savlen1 = 0;                                ! Use savelen1 as 'flag' to see if saved
;    1095  2          len1 = .key1 [dsc$w_length];
;    1096  2          len2 = .key2 [dsc$w_length];
;    1097  2          ptr1 = .key1 [dsc$a_pointer];
;    1098  2          ptr2 = .key2 [dsc$a_pointer];
;    1099  2
;    1100  2          WHILE 1 DO
;    1101  2      loop :
;    1102  3              BEGIN
;    1103  3
;    1104  3              LOCAL
;    1105  3                  char : BYTE;
;    1106  3
;    1107  3              len2 = .len2 - 1;
;    1108  3
;    1109  3              IF .len2 LSS 0
;    1110  3              THEN
;    1111  3
;    1112  3                  IF .len1 EQL 0
;    1113  3                  THEN
;    1114  3                      RETURN 1
;    1115  3                  ELSE
;    1116  4                      BEGIN
;    1117  4
;    1118  4                      IF (savlen1 = .savlen1 - 1) LSS 0       ! Partial match
;    1119  4                      THEN
;    1120  4
;    1121  4                          IF .wild
;    1122  4                          THEN
;    1123  4                              RETURN 0                ! Don't take partial, if wild
;    1124  4                          ELSE
;    1125  5                              RETURN (.wildcompare = 1)       ! Partial OK, elsewise
;    1126  4                      ELSE
;    1127  4
;    1128  5                          BEGIN                       ! Restore saved valued--backup
;    1129  5                          ptr1 = (savptr1 = .savptr1 + 1);    ! Advance to next char.
```

```
;    1130  5                        ptr2 = .savptr2;
;    1131  5                        len1 = .savlen1;
;    1132  5                        len2 = .savlen2
;    1133  4                        END;
;    1134  4
;    1135  4                    LEAVE loop                          ! Proceed to next iteration of loop
;    1136  3                    END;
;    1137  3
;    1138  3                char = CH$RCHAR_A (ptr2);               ! Get next pattern character
;    1139  3
;    1140  3                IF .char EQL %C'*'
;    1141  3                THEN
;    1142  4                    BEGIN                               ! If wildcard char, save descriptors for later backup.
;    1143  4                    wild = 1;                           ! Set wildcard flag
;    1144  4
;    1145  4                    IF .len2 EQL 0 THEN RETURN (.wildcompare = 1);      ! '*' at end matches all
;    1146  4
;    1147  4                    savptr1 = .ptr1;
;    1148  4                    savptr2 = .ptr2;
;    1149  4                    savlen1 = .len1;
;    1150  4                    savlen2 = .len2
;    1151  4                    END
;    1152  3                ELSE
;    1153  3
;    1154  3                    IF
;    1155  4                        BEGIN
;    1156  4                        len1 = .len1 - 1;
;    1157  5                        BEGIN
;    1158  5
;    1159  5                        LOCAL
;    1160  5                            ch : BYTE;
;    1161  5
;    1162  5                        ch = CH$RCHAR_A (ptr1);
;    1163  7                        (.ch = (IF .ch GEQU %C'a' AND .ch LEQU %C'z'    ! convert to upper case
;    1164  6                        THEN %C'a' - %C'A' ELSE 0))
;    1165  5                        END
;    1166  4                        END
;    1167  3                        NEQ .char                       ! If not a match
;    1168  3                    THEN
;    1169  3
;    1170  3                        IF .char NEQ %C'%'              ! and not a character wildcard
;    1171  3                        THEN
;    1172  3
;    1173  3                            IF (savlen1 = .savlen1 - 1) LSS 0
;    1174  3                            THEN
;    1175  3                                RETURN 0
;    1176  3                            ELSE
;    1177  4                                BEGIN                   ! Restore values
;    1178  4                                ptr1 = (savptr1 = .savptr1 + 1);        ! Advance to next char
;    1179  4                                ptr2 = .savptr2;
;    1180  4                                len1 = .savlen1;
;    1181  4                                len2 = .savlen2
;    1182  4                                END
;    1183  4
;    1184  3                        ELSE
;    1185  3                            .wildcompare = 1
;    1186  3
```

```
;   1187  2              END;                          ! loop
;   1188  2
;   1189  2          0                                 ! If we leave loop, no good.
;   1190  1          END;                              ! of Dsr$Key_Equal



                                      0FFC 00000          .ENTRY  DSR$KEY_EQUAL, Save R2,R3,R4,R5,R6,R7,R8,-  ; 1045
                                                                  R9,R10,R11                                      ;
                                5E        04  C2 00002          SUBL2   #4, SP                                  ; 1093
                                          5B  D4 00005          CLRL    WILD                                    ;
                                   0C     BC  D4 00007          CLRL    @WILDCOMPARE                            ; 1094
                                          53  D4 0000A          CLRL    SAVLEN1                                 ; 1095
                                51        04  AC  D0 0000C      MOVL    KEY1, R1                                ;
                                55        61  3C 00010        MOVZWL  (R1), LEN1                              ; 1096
                                50        08  AC  D0 00013      MOVL    KEY2, R0                                ;
                                54        60  3C 00017        MOVZWL  (R0), LEN2                              ; 1097
                                58        04  A1  D0 0001A      MOVL    4(R1), PTR1                             ; 1098
                                57        04  A0  D0 0001E      MOVL    4(R0), PTR2                             ; 1107
                                0C            54  F4 00022 1$:  SOBGEQ  LEN2, 2$                               ; 1112
                                              55  D5 00025      TSTL    LEN1                                    ;
                                              1B  13 00027      BEQL    4$                                      ; 1118
                                5B            53  F4 00029      SOBGEQ  SAVLEN1, 9$                             ; 1121
                                11            5B  E9 0002C      BLBC    WILD, 3$                                ; 1125
                                              6C  11 0002F      BRB     11$                                     ; 1138
                                52            87  90 00031 2$: MOVB    (PTR2)+, CHAR                           ; 1140
                                2A            52  91 00034      CMPB    CHAR, #42                               ;
                                              1D  12 00037      BNEQ    6$                                      ; 1143
                                5B            01  D0 00039      MOVL    #1, WILD                                ; 1145
                                              54  D5 0003C      TSTL    LEN2                                    ;
                                              08  12 0003E      BNEQ    5$                                      ;
                             0C   BC          01  D0 00040 3$:  MOVL    #1, @WILDCOMPARE                        ;
                                50            01  D0 00044 4$:  MOVL    #1, R0                                  ;
                                              04  00047         RET                                             ;
                                56            58  D0 00048 5$:  MOVL    PTR1, SAVPTR1                           ; 1147
                                59            57  D0 0004B      MOVL    PTR2, SAVPTR2                           ; 1148
                                53            55  D0 0004E      MOVL    LEN1, SAVLEN1                           ; 1149
                                5A            54  D0 00051      MOVL    LEN2, SAVLEN2                           ; 1150
                                              CC  11 00054      BRB     1$                                      ;
                                              55  D7 00056 6$:  DECL    LEN1                                    ; 1156
                                51            88  90 00058      MOVB    (PTR1)+, CH                             ; 1162
                             61   8F          51  91 0005B      CMPB    CH, #97                                 ; 1163
                                              0B  1F 0005F      BLSSU   7$                                      ;
                             7A   8F          51  91 00061      CMPB    CH, #122                                ;
                                              05  1A 00065      BGTRU   7$                                      ;
                                50            20  D0 00067      MOVL    #32, R0                                 ; 1164
                                              02  11 0006A      BRB     8$                                      ;
                                              50  D4 0006C 7$:  CLRL    R0                                      ; 1163
                                6E            51  9A 0006E 8$:  MOVZBL  CH, (SP)                               ;
                                50            6E  C2 00071      SUBL2   (SP), R0                                ;
                                50            50  CE 00074      MNEGL   R0, R0                                  ;
50                52            08            00  ED 00077      CMPZV   #0, #8, CHAR, R0                        ; 1167
                                              A4  13 0007C      BEQL    1$                                      ;
                                25            52  91 0007E      CMPB    CHAR, #37                               ; 1170
                                              14  13 00081      BEQL    10$                                     ;
```

```
                                                53  D7 00083          DECL    SAVLEN1                                 ; 1173
                                                16  19 00085          BLSS    11$                                     ;
                                                56  D6 00087 9$:      INCL    SAVPTR1                                 ; 1178
                                          58    56  D0 00089          MOVL    SAVPTR1, PTR1                           ;
                                          57    59  D0 0008C          MOVL    SAVPTR2, PTR2                           ; 1179
                                          55    53  D0 0008F          MOVL    SAVLEN1, LEN1                           ; 1180
                                          54    5A  D0 00092          MOVL    SAVLEN2, LEN2                           ; 1181
                                                8B  11 00095          BRB     1$                                      ; 1173
                                       0C BC    01  D0 00097 10$:     MOVL    #1, @WILDCOMPARE                        ; 1185
                                                85  11 0009B          BRB     1$                                      ; 1170
                                                50  D4 0009D 11$:     CLRL    R0                                      ; 1190
                                                    04 0009F          RET                                             ;

; Routine Size:  160 bytes,    Routine Base:  CODE + 07A5

;   1191  1
```

```
;    1192  1     %SBTTL 'skip_blanks'
;    1193  1     ROUTINE skip_blanks (desc) =
;    1194  1
;    1195  1     !++
;    1196  1     ! FUNCTIONAL DESCRIPTION
;    1197  1     !       Advance line descriptor past blanks
;    1198  1     !
;    1199  1     ! INPUTS
;    1200  1     !       desc    pointer to descriptor of text
;    1201  1     !
;    1202  1     ! IMPLICIT INPUTS
;    1203  1     !       none
;    1204  1     !
;    1205  1     ! OUTPUTS
;    1206  1     !       desc    descriptor is updated past blanks
;    1207  1     !
;    1208  1     ! IMPLICIT OUTPUTS
;    1209  1     !       none
;    1210  1     !
;    1211  1     ! SIDE EFFECTS
;    1212  1     !       none
;    1213  1     !
;    1214  1     ! RETURN CODES
;    1215  1     !       0       if end-of-line detected
;    1216  1     !       1       if non-blank character found
;    1217  1     !--
;    1218  1
```

```
;    1219  2          BEGIN
;    1220  2
;    1221  2          MAP
;    1222  2              desc : REF bblock;
;    1223  2
;    1224  2          LOCAL
;    1225  2              char : BYTE;
;    1226  2
;    1227  2          WHILE .desc [dsc$w_length] GTR 0 DO
;    1228  3              BEGIN
;    1229  3              char = CH$RCHAR (.desc [dsc$a_pointer]);        ! Get next character
;    1230  3
;    1231  3              IF .char EQL %C'!' THEN RETURN 0;        ! Comment delimits line
;    1232  3
;    1233  3              IF .char NEQ %C' ' AND .char NEQ %C'    ' THEN RETURN 1;        ! Done
;    1234  3
;    1235  3              desc [dsc$a_pointer] = .desc [dsc$a_pointer] + 1;        ! Advance descriptor
;    1236  3              desc [dsc$w_length] = .desc [dsc$w_length] - 1
;    1237  2              END;
;    1238  2
;    1239  2          0                                               ! Hit end of line
;    1240  1          END;                                            ! of skip_blanks


                                        0000 00000 SKIP_BLANKS:
                                                   .WORD   Save nothing                    ; 1193
                50      04   AC  D0 00002          MOVL    DESC, R0                        ; 1227
                             60  B5 00006 1$:      TSTW    (R0)                            ;
                             1E  13 00008          BEQL    3$                              ;
                51      04   B0  90 0000A          MOVB    @4(R0), CHAR                    ; 1229
                21           51  91 0000E          CMPB    CHAR, #33                       ; 1231
                             15  13 00011          BEQL    3$                              ;
                20           51  91 00013          CMPB    CHAR, #32                       ; 1233
                             09  13 00016          BEQL    2$                              ;
                09           51  91 00018          CMPB    CHAR, #9                        ;
                             04  13 0001B          BEQL    2$                              ;
                50           01  D0 0001D          MOVL    #1, R0                          ;
                             04  00020             RET                                     ;
                        04   A0  D6 00021 2$:      INCL    4(R0)                           ; 1235
                             60  B7 00024          DECW    (R0)                            ; 1236
                             DE  11 00026          BRB     1$                              ;
                             50  D4 00028 3$:      CLRL    R0                              ; 1240
                                 04 0002A          RET                                     ;

; Routine Size:  43 bytes,    Routine Base:  CODE + 0845
```

```
;    1241  1     %SBTTL 'scan_word'
;    1242  1     ROUTINE scan_word (desc) =
;    1243  1
;    1244  1     !++
;    1245  1     ! FUNCTIONAL DESCRIPTION
;    1246  1     !       scan descriptor over a word
;    1247  1     !
;    1248  1     ! INPUTS
;    1249  1     !       desc    address of descriptor for string
;    1250  1     !
;    1251  1     ! IMPLICIT INPUTS
;    1252  1     !       none
;    1253  1     !
;    1254  1     ! OUTPUTS
;    1255  1     !       desc    descriptor is updated past word scanned
;    1256  1     !       R0      length of word scanned
;    1257  1     !
;    1258  1     ! IMPLICIT OUTPUTS
;    1259  1     !       none
;    1260  1     !
;    1261  1     ! SIDE EFFECTS
;    1262  1     !       none
;    1263  1     !
;    1264  1     ! RETURN CODES
;    1265  1     !       none    (R0 if function value)
;    1266  1     !--
;    1267  1
```

```
;    1268  2          BEGIN
;    1269  2
;    1270  2          MAP
;    1271  2              desc : REF bblock;
;    1272  2
;    1273  2          LOCAL
;    1274  2              firstchar,                              ! flag
;    1275  2              startpointer;                           ! Point to initial character
;    1276  2
;    1277  2          startpointer = .desc [dsc$a_pointer];
;    1278  2          firstchar = 1;                              ! On first character
;    1279  2
;    1280  2          WHILE .desc [dsc$w_length] GTR 0 DO         ! Scan string
;    1281  3              BEGIN
;    1282  3
;    1283  3              LOCAL
;    1284  3                  char : BYTE;
;    1285  3
;    1286  3              char = CH$RCHAR (.desc [dsc$a_pointer]);        ! Read next character
;    1287  3              !
;    1288  3              ! If it's not a legal character to have inside a word, exit
;    1289  3              !
;    1290  3
;    1291  5              IF NOT ((.char GEQU %C'A' AND .char LEQU %C'Z') OR (.char GEQU %C'a' AND .char LEQU %C'z') OR (.char GEQU
;    1292  5                  %C'0' AND .char LEQU %C'9') OR (.char EQL %C'$')     ! dollar sign
;    1293  5                  OR (.char EQL %C'_')                ! underscore
;    1294  5                  OR (.char EQL %C'-')                ! hyphen
;    1295  5                  OR (.char EQL %C'.')                ! period
;    1296  5                  OR (.char EQL %C'%')                ! wildcard
;    1297  5                  OR (.char EQL %C'*')                ! wildcard
;    1298  4                  OR (.firstchar AND (.char EQL %C'/')))      ! slash (for qualifier)
;    1299  3              THEN
;    1300  3                  EXITLOOP;                           ! Exit if done with word
;    1301  3
;    1302  3              firstchar = 0;                          ! Reset flag
;    1303  3              desc [dsc$a_pointer] = .desc [dsc$a_pointer] + 1;
;    1304  3              desc [dsc$w_length] = .desc [dsc$w_length] - 1
;    1305  2              END;
;    1306  2
;    1307  3          (CH$DIFF (.desc [dsc$a_pointer], .startpointer))
;    1308  1          END;                                        ! of scan_word


                                        000C 00000 SCAN_WORD:
                                                           .WORD    Save R2,R3                          ;  1242
                              50        04     AC  D0 00002        MOVL     DESC, R0                    ;  1277
                              53        04     A0  D0 00006        MOVL     4(R0), STARTPOINTER         ;
                              52               01  D0 0000A        MOVL     #1, FIRSTCHAR               ;  1278
                                               60  B5 0000D 1$:    TSTW     (R0)                        ;  1280
                                               56  13 0000F        BEQL     6$                          ;
                              51        04     B0  90 00011        MOVB     @4(R0), CHAR                ;  1286
                        41    8F               51  91 00015        CMPB     CHAR, #65                   ;  1291
                                               06  1F 00019        BLSSU    2$                          ;
                        5A    8F               51  91 0001B        CMPB     CHAR, #90                   ;
```

```
                                        3D  1B 0001F          BLEQU   5$                                      ;
                                61   8F  51  91 00021 2$:     CMPB    CHAR, #97                               ;
                                        06  1F 00025          BLSSU   3$                                      ;
                                7A   8F  51  91 00027         CMPB    CHAR, #122                              ;
                                        31  1B 0002B          BLEQU   5$                                      ;
                                30       51  91 0002D 3$:     CMPB    CHAR, #48                               ;
                                        05  1F 00030          BLSSU   4$                                      ;
                                39       51  91 00032         CMPB    CHAR, #57                               ; 1292
                                        27  1B 00035          BLEQU   5$                                      ;
                                24       51  91 00037 4$:     CMPB    CHAR, #36                               ;
                                        22  13 0003A          BEQL    5$                                      ;
                                5F   8F  51  91 0003C         CMPB    CHAR, #95                               ; 1293
                                        1C  13 00040          BEQL    5$                                      ;
                                2D       51  91 00042         CMPB    CHAR, #45                               ; 1294
                                        17  13 00045          BEQL    5$                                      ;
                                2E       51  91 00047         CMPB    CHAR, #46                               ; 1295
                                        12  13 0004A          BEQL    5$                                      ;
                                25       51  91 0004C         CMPB    CHAR, #37                               ; 1296
                                        0D  13 0004F          BEQL    5$                                      ;
                                2A       51  91 00051         CMPB    CHAR, #42                               ; 1297
                                        08  13 00054          BEQL    5$                                      ;
                                0E       52  E9 00056         BLBC    FIRSTCHAR, 6$                           ; 1298
                                2F       51  91 00059         CMPB    CHAR, #47                               ;
                                        09  12 0005C          BNEQ    6$                                      ;
                                        52  D4 0005E 5$:      CLRL    FIRSTCHAR                               ; 1302
                            04  A0  D6 00060                  INCL    4(R0)                                   ; 1303
                                60  B7 00063                  DECW    (R0)                                    ; 1304
                                A6  11 00065                  BRB     1$                                      ;
                50      04  A0      53  C3 00067 6$:         SUBL3   STARTPOINTER, 4(R0), R0                 ; 1307
                                        04 0006C              RET                                             ; 1308

; Routine Size:  109 bytes,    Routine Base:  CODE + 0870
```

```
;    1309  1      %SBTTL 'find character'
;    1310  1      ROUTINE find_char (desc, character) : jsb_2 =
;    1311  1
;    1312  1      !++
;    1313  1      ! FUNCTIONAL DESCRIPTION
;    1314  1      !       find distance to character in string
;    1315  1      !
;    1316  1      ! INPUTS
;    1317  1      !       desc            address of string descriptor to search
;    1318  1      !       character       the character to look for
;    1319  1      !
;    1320  1      ! IMPLICIT INPUTS
;    1321  1      !       none
;    1322  1      !
;    1323  1      ! OUTPUTS
;    1324  1      !       R0              distance in string to character
;    1325  1      !
;    1326  1      ! IMPLICIT OUTPUTS
;    1327  1      !       none
;    1328  1      !
;    1329  1      ! SIDE EFFECTS
;    1330  1      !       none
;    1331  1      !
;    1332  1      ! RETURN CODES
;    1333  1      !       R0 is function value
;    1334  1      !--
;    1335  1
```

```
;    1336  2          BEGIN
;    1337  2
;    1338  2          LOCAL
;    1339  2              ptr,
;    1340  2              len;
;    1341  2
;    1342  2          MAP
;    1343  2              desc : REF bblock,
;    1344  2              character : BYTE;
;    1345  2
;    1346  2          len = .desc [dsc$w_length];
;    1347  2          ptr = .desc [dsc$a_pointer];
;    1348  2
;    1349  2          WHILE .len GTR 0 DO
;    1350  3              BEGIN
;    1351  3              len = .len - 1;
;    1352  3
;    1353  3              IF CH$RCHAR (.ptr) EQL .character THEN EXITLOOP;        ! If we found it
;    1354  3
;    1355  3              ptr = .ptr + 1
;    1356  2              END;
;    1357  2
;    1358  2          CH$DIFF (.ptr, .desc [dsc$a_pointer])        ! Return distance
;    1359  1          END;                                         ! of find_char



                                 0C  BB 00000 FIND_CHAR:
                                                  PUSHR   #^M<R2,R3>                      ; 1310
                          53         60  3C 00002          MOVZWL  (DESC), LEN             ; 1346
                          52    04   A0  D0 00005          MOVL    4(DESC), PTR            ; 1347
                                     53  D5 00009 1$:      TSTL    LEN                     ; 1349
                                     0B  15 0000B          BLEQ    2$                      ;
                                     53  D7 0000D          DECL    LEN                     ; 1351
                          51         62  91 0000F          CMPB    (PTR), CHARACTER        ; 1353
                                     04  13 00012          BEQL    2$                      ;
                                     52  D6 00014          INCL    PTR                     ; 1355
                                     F1  11 00016          BRB     1$                      ;
                     50   52    04   A0  C3 00018 2$:      SUBL3   4(DESC), PTR, R0        ; 1358
                                     0C  BA 0001D          POPR    #^M<R2,R3>              ; 1359
                                         05 0001F          RSB                             ;

; Routine Size:  32 bytes,    Routine Base:  CODE + 08DD
```

```
;    1360  1      %SBTTL 'print blank line'
;    1361  1      ROUTINE print_crlf : jsb_none NOVALUE =
;    1362  1
;    1363  1      !++
;    1364  1      ! Print a blank line
;    1365  1      !--
;    1366  1
;    1367  2          BEGIN
;    1368  3          $ds_printf ($ascic ('!/'))
;    1369  1          END;                                        ! of print_crlf


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                        2F  21  02  000A4 P.AAL:  .ASCII  <2>\!/\                                   ;


                                                                .PSECT  CODE,NOWRT,  SHR,2

                                00000000'  EF  9F 00000 PRINT_CRLF:
                                                                PUSHAB  P.AAL                                       ; 1368
                        00000000G  9F           01  FB 00006    CALLS   #1, @#DS$PRINTF                             ;
                                                    05 0000D    RSB                                                 ; 1369

; Routine Size:  14 bytes,    Routine Base:  CODE + 08FD
```

```
;   1370  1     END
;   1371  1
;   1372  0     ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  DATA                           167  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                            46  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          2315  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                                 Library Statistics
;
;                                  -------- Symbols --------      Pages       Processing
;       File                       Total    Loaded   Percent      Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265        784         3         0          85        00:00.1
;  DRB1:[DS.WORK]DS.L32;159          653        19         2          42        00:00.2
;  SYS$SYSROOT:[SYSLIB]STARLET.L32;60 9486       76         0         570        00:02.6
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE HELP

; Size:          2315 code + 213 data bytes
; Run Time:         00:43.6
; Elapsed Time:     01:55.1
; Lines/CPU Min:     1887
; Lexemes/CPU-Min: 18909
; Memory Used:  235 pages
; Compilation Complete
```

Table of contents

(2) 108 DECLARATIONS
(3) 199 Device description database

```
0000     1          .TITLE  ICODE - Interpreted code for TSP
0000     2          .IDENT  /07-22/
0000     3          .LIST   MEB
0000     4          .NLIST  CND
0000     5
0000     6 ;
0000     7 ; Copyright (c) 1979, 1982, 1983
0000     8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     9 ;
0000    10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    15 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    16 ; REMAIN IN DEC.
0000    17 ;
0000    18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    20 ; CORPORATION.
0000    21 ;
0000    22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    24
0000    25 ;++
0000    26 ; FACILITY:     TSP
0000    27 ;
0000    28 ; ABSTRACT:     This module is used to generate the necessary driver
0000    29 ;               for the APT Attach command interpreter.  It is really
0000    30 ;               part of TSP.
0000    31 ;
0000    32 ; ENVIRONMENT:
0000    33 ;
0000    34 ; AUTHOR:       R. Riggs        15-Jul-1979
0000    35 ;
0000    36 ; MODIFIED BY:
0000    37 ;               Dave Butenhof, 2-apr-1981 (with Supervisor V6.3)
0000    38 ;       01      Add PTdescriptors for ML11, KA730, LP25, TM78 and TU78
0000    39 ;
0000    40 ;       02      Dave Butenhof, 13-Nov-1981, version 6.5
0000    41 ;               Add Ptable descriptors for DW730, DMF32[A,P,S, ], UDA50,
0000    42 ;               RB730, R80, and RA80.
0000    43 ;
0000    44 ;       [03] Dave butenhof, 09-Apr-1982, version 6.7
0000    45 ;               Support $DS_$NAME directive (ignore it)
0000    46 ;       [04] Dave Butenhof, 03-May-1982, version 6.7
0000    47 ;               Add RA81 and RA60 devices
0000    48 ;
0000    49 ;       05      M. Baggett       24-Sept-1982 Version 6.9
0000    50 ;               added CI750
0000    51 ;
0000    52 ;       06      Bob Bergazzi    2-Nov-1982      Version 6.9
0000    53 ;               added UNA11
0000    54 ;
0000    55 ;       07      M. Baggett       3-Nov-1982     Version 6.9
0000    56 ;               Added CI750 macro call.
0000    57 ;
```

```
                                0000     58 ;        08      M. BAGGETT      1-Dec-1982      Version 6.10
                                0000     59 ;                Added TU80 and DZ32.
                                0000     60 ;
                                0000     61 ;        09      M. BAGGETT      8-DEC-1982      Version 6.10
                                0000     62 ;                Added TU81.
                                0000     63 ;
                                0000     64 ;        10      M. BAGGETT      10-DEC-1982     Version 6.10
                                0000     65 ;                Added VS100.
                                0000     66 ;
                                0000     67 ;        11      M. BAGGETT      14-JAN-1983     Version 6.11
                                0000     68 ;                Added RC25.
                                0000     69 ;
                                0000     70 ;        12      M. Baggett      23-Feb-1983     Version 6.11
                                0000     71 ;                Added CI_NODE and RCF25,VS300.
                                0000     72 ;
                                0000     73 ;        13      M. Baggett      14-Mar-1983     Version 6.11
                                0000     74 ;                Added p-table for LESI controller.
                                0000     75 ;
                                0000     76 ;        14      M. Baggett      21-Mar-1983     Version 6.11
                                0000     77 ;                Added VT220,VT240,LA12,LA100,LN01.
                                0000     78 ;
                                0000     79 ;        15      Richard Brown   18-July-83      Version 6.12
                                0000     80 ;                Added CONSOLE, KAXXX, KAYYY, LP07, LP26, LP27,
                                0000     81 ;                RC11, SBIA.
                                0000     82 ;
                                0000     83 ;        16      M. Baggett      8-Aug-1983      Version 6.12
                                0000     84 ;                Added IEU11A.
                                0000     85 ;
                                0000     86 ;        17      M. Baggett      7-Sep-1983      Version 6.13
                                0000     87 ;                Added VS125.
                                0000     88 ;
                                0000     89 ;        18      M. Baggett      22-Nov-1983     Version 6.13
                                0000     90 ;                Added TS05,DISK.
                                0000     91 ;
                                0000     92 ;        19      Richard Brown   5-Mar-84        Version 6.14
                                0000     93 ;                Added DEQNA, DLVJ1, DZV11, RQDX1, RX50, RD51, RD52
                                0000     94 ;                for MicroVAX I.
                                0000     95 ;
                                0000     96 ;        20      Richard Brown   3-Apr-84        Version 7.0
                                0000     97 ;                Corrected typo which included RD51 twice and
                                0000     98 ;                excluded RD52.  Now both RD51 and  RD52  are
                                0000     99 ;                there.
                                0000    100 ;
                                0000    101 ;        21      M. Baggett      21-Jun-84       Version 7.0
                                0000    102 ;                Added DMZ32.
                                0000    103 ;
                                0000    104 ;        22      Richard Brown   7-Jul-84        Version 7.0
                                0000    105 ;                Removed RC11, since it is really LESI.
                                0000    106 ;--
```

```
                        0000   108          .SBTTL  DECLARATIONS
                        0000   109 ;
                        0000   110 ; INCLUDE FILES:
                        0000   111 ;
                        0000   112
                        0000   113 .LIBRARY        /$DIAG/
                        0000   114
                        0000   115 ;
                        0000   116 ; MACROS:
                        0000   117 ;
                        0000   118          $DS_HPODEF
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
                00000008 0000              .IIF    NB,.BLKQ,       .BLKQ   1
                        0008               .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
                0000000A 0008              .IIF    NB,.BLKQ,       .BLKQ   1
                        000A               .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
                0000000B 000A              .IIF    NB,.BLKB,       .BLKB   1
                        000B               .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
                0000000C 000B              .IIF    NB,.BLKB,       .BLKB   1
                        000C               .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
                00000018 000C              .IIF    NB,.BLKB,       .BLKB   12
                        0018               .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
                0000001C 0018              .IIF    NB,.BLKL,       .BLKL   1
                        001C               .IIF    NB,HP$A_DVA,    HP$A_DVA:
                00000020 001C              .IIF    NB,.BLKL,       .BLKL   1
                        0020               .IIF    NB,HP$A_LINK,   HP$A_LINK:
                00000024 0020              .IIF    NB,.BLKL,       .BLKL   1
                        0024               .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
                00000026 0024              .IIF    NB,.BLKQ,       .BLKQ   1
                        0026               .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
                00000032 0026              .IIF    NB,.BLKB,       .BLKB   12
                        0032               .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                        0000               .RESTORE
                        0000   119
                        0000   120 .MACRO   $DS_$INITIALIZE DEVICE,LENGTH,MAX,DRIVER
                        0000   121          .ASCIC  "DEVICE"            ; Device name
                        0000   122          .BYTE   MAX                 ; Maximum unit number
                        0000   123          .BYTE   ^X80                ; Constant to identify start of descriptor
                        0000   124 .ENDM    $DS_$INITIALIZE
                        0000   125
                        0000   126 .MACRO   $DS_$NAME       FLAGS, GENERIC
                        0000   127 .ENDM    $DS_$NAME
                        0000   128
                        0000   129 .MACRO   $DS_$END
                        0000   130          .BYTE   ^X81                ; Indicate end of PT-descriptor
                        0000   131 .ENDM    $DS_$END
                        0000   132
                        0000   133 .MACRO   $DS_$DECIMAL    PROMPT,LOW,HIGH
                        0000   134          .BYTE   ^X82                ; Indicate scan decimal
                        0000   135          .ASCIC  "PROMPT"            ; prompt string
                        0000   136          .LONG   LOW,HIGH            ; Low , High limits on input
                        0000   137 .ENDM    $DS_$DECIMAL
                        0000   138
                        0000   139 .MACRO   $DS_$OCTAL      PROMPT,LOW,HIGH
                        0000   140          .BYTE   ^X83                ; Indicate scan octal
                        0000   141          .ASCIC  "PROMPT"            ; prompt string
```

```
0000   142          .LONG   ^O'LOW,^O'HIGH  ; Low , High limits on input
0000   143 .ENDM    $DS_$OCTAL
0000   144
0000   145 .MACRO   $DS_$HEX          PROMPT,LOW,HIGH
0000   146          .BYTE   ^X84                ; Indicate scan hexadecimal
0000   147          .ASCIC  "PROMPT"            ; prompt string
0000   148          .LONG   LOW,HIGH            ; Low , High limits on input
0000   149 .ENDM    $DS_$HEX
0000   150
0000   151 .MACRO   $DS_$STRING       PROMPT,STRINGS
0000   152          .BYTE   ^X85                ; Indicate scan and verify string
0000   153          .ASCIC  "PROMPT"            ; prompt string
0000   154          .IRP    STRING,STRINGS
0000   155          .ASCIC  "STRING"
0000   156          .ENDR
0000   157          .BYTE   0                   ; Null length is end of valid strings
0000   158 .ENDM    $DS_$STRING
0000   159
0000   160 .MACRO   $DS_$LOGICAL      PROMPT
0000   161          .BYTE   ^X85                ; For APT, pretend this is a STRING
0000   162          .ASCIC  "PROMPT"
0000   163          .ASCIC  "NO"
0000   164          .ASCIC  "YES"
0000   165          .BYTE   0
0000   166 .ENDM    $DS_$LOGICAL
0000   167
0000   168 .MACRO   $DS_$LITERAL      ARG
0000   169 $$$$$$$ = ARG
0000   170 .ENDM    $DS_$LITERAL
0000   171
0000   172 .MACRO   $DS_$FETCH        OFFSET,BIT,SIZE
0000   173 .ENDM    $DS_$FETCH
0000   174
0000   175 .MACRO   $DS_$STORE        OFFSET,BIT,SIZE
0000   176 .ENDM    $DS_$STORE
0000   177
0000   178 .MACRO   $DS_$COMPLEMENT
0000   179 .ENDM    $DS_$COMPLEMENT
0000   180
0000   181 .MACRO   $DS_$ADD          OFFSET,BIT,SIZE
0000   182 .ENDM    $DS_$ADD
0000   183
0000   184 .MACRO   $DS_$CASE         CASEPAIRS
0000   185 .ENDM    $DS_$CASE
0000   186
0000   187 .MACRO   $DS_DEVTYP        STRINGS,ADDRESSES
0000   188 $$N=0
0000   189 .IRP X,ADDRESSES
0000   190 $$N=$$N+1
0000   191 .ENDR
0000   192 AL_DEVTYP:
0000   193          .WORD   $$N
0000   194 .IRP X,ADDRESSES
0000   195          .WORD   X-AL_DEVTYP
0000   196 .ENDR
0000   197 .ENDM    $DS_DEVTYP
```

```
                     0000   199          .SBTTL  Device description database
                     0000   200 ;
                     0000   201 ;       The following list contains the descriptor for each type
                     0000   202 ;       of device the supervisor known about.
                     0000   203 ;
                     0000   204
                     0000   205 DS$GA_PTDESC::
                     0000   206          $DS_DEVTYP      <>,<AA11K,AD11K,CI_NODE,CI780,CI750,CONSOLE,-          ;
                     0000   207                          CR11,DEQNA,DISK,DLT1,DLVJ1 -                           ;
                     0000   208                          DMC11,DMF32,DMF32A,DMF32P,DMF32S,-                     ;
                     0000   209                          DMP11,DMR11,DMZ32,DR11B,-                              ;  [21]
                     0000   210                          DR11K,DR11W,DR750,DR780, -
                     0000   211                          DUP11,DW730,DW750,DW780,DZ11,DZ32,DZV11,IEU11A,-        ;
                     0000   212                          KA730,KA750,KA780,KAXXX,KA785, -                       ;
                     0000   213                          KMC11,KW11K,LA12,LA34,LA36,LA38, -                     ;  [14]
                     0000   214                          LA100,LA120,LA180,LESI,LN01,LP04, -                    ;  [13]
                     0000   215                          LP05,LP06,LP07,LP11,LP14, -                            ;  [15]
                     0000   216                          LP25,LP26,LP27,LPA11K,MA780,MBE,-                      ;  [01]
                     0000   217                          ML11,MS750,MS780,PCL11,-                               ;  [01]
                     0000   218                          PX780,-                                                ;
                     0000   219                          R80,RA60,RA80,RA81,RB730,RCF25,RC25, -                 ;  [12]
                     0000   220                          RD51,RD52,RH750,RH780,RL01,RL02, -                     ;  [19]
                     0000   221                          RL11,RK06,RK07,RK611, -
                     0000   222                          RM03,RM05,RM80, -
                     0000   223                          RP04,RP05,RP06,RP07,RQDX1,RX02, -                      ;  [19]
                     0000   224                          RX211,RX50,SBIA,TE16,TM03,TM78,-                       ;  [01]
                     0000   225                          TS04,TS05,TS11,TU45,TU58,TU77,TU78,TU80,TU81-          ;  [18]
                     0000   226                          UBE,UDA50,UNA11,-                                      ;  [06]
                     0000   227                          VS100,VS125,VS300,VT50,VT52,VT55,VT100,-               ;  [12]
                     0000   228                          VT220,VT240>
                     0000        AL_DEVTYP:
0076  0000                              .WORD   $$N
00EE' 0002                              .WORD   AA11K-AL_DEVTYP
0128' 0004                              .WORD   AD11K-AL_DEVTYP
0162' 0006                              .WORD   CI_NODE-AL_DEVTYP
01E1' 0008                              .WORD   CI780-AL_DEVTYP
01B0' 000A                              .WORD   CI750-AL_DEVTYP
0210' 000C                              .WORD   CONSOLE-AL_DEVTYP
021B' 000E                              .WORD   CR11-AL_DEVTYP
024C' 0010                              .WORD   DEQNA-AL_DEVTYP
0262' 0012                              .WORD   DISK-AL_DEVTYP
026A' 0014                              .WORD   DL11-AL_DEVTYP
029B' 0016                              .WORD   DLVJ1-AL_DEVTYP
0334' 0018                              .WORD   DMC11-AL_DEVTYP
0366' 001A                              .WORD   DMF32-AL_DEVTYP
0398' 001C                              .WORD   DMF32A-AL_DEVTYP
0497' 001E                              .WORD   DMF32P-AL_DEVTYP
04E6' 0020                              .WORD   DMF32S-AL_DEVTYP
0539' 0022                              .WORD   DMP11-AL_DEVTYP
05C7' 0024                              .WORD   DMR11-AL_DEVTYP
05F9' 0026                              .WORD   DMZ32-AL_DEVTYP
0725' 0028                              .WORD   DR11B-AL_DEVTYP
0757' 002A                              .WORD   DR11K-AL_DEVTYP
0791' 002C                              .WORD   DR11W-AL_DEVTYP
07C3' 002E                              .WORD   DR750-AL_DEVTYP
0824' 0030                              .WORD   DR780-AL_DEVTYP
0883' 0032                              .WORD   DUP11-AL_DEVTYP
```

```
                        08E6' 0034              .WORD   DW730-AL_DEVTYP
                        08EF' 0036              .WORD   DW750-AL_DEVTYP
                        08F8' 0038              .WORD   DW780-AL_DEVTYP
                        0919' 003A              .WORD   DZ11-AL_DEVTYP
                        0961' 003C              .WORD   DZ32-AL_DEVTYP
                        0A1D' 003E              .WORD   DZV11-AL_DEVTYP
                        0A43' 0040              .WORD   IEU11A-AL_DEVTYP
                        0A77' 0042              .WORD   KA730-AL_DEVTYP
                        0B1C' 0044              .WORD   KA750-AL_DEVTYP
                        0BB7' 0046              .WORD   KA780-AL_DEVTYP
                        0C36' 0048              .WORD   KAXXX-AL_DEVTYP
                        0CB5' 004A              .WORD   KA785-AL_DEVTYP
                        0D34' 004C              .WORD   KMC11-AL_DEVTYP
                        0D66' 004E              .WORD   KW11K-AL_DEVTYP
                        0DA0' 0050              .WORD   LA12-AL_DEVTYP
                        0DC3' 0052              .WORD   LA34-AL_DEVTYP
                        0DCB' 0054              .WORD   LA36-AL_DEVTYP
                        0DD3' 0056              .WORD   LA38-AL_DEVTYP
                        0DA8' 0058              .WORD   LA100-AL_DEVTYP
                        0DB1' 005A              .WORD   LA120-AL_DEVTYP
                        0DBA' 005C              .WORD   LA180-AL_DEVTYP
                        0DDB' 005E              .WORD   LESI-AL_DEVTYP
                        0E0B' 0060              .WORD   LN01-AL_DEVTYP
                        0E13' 0062              .WORD   LP04-AL_DEVTYP
                        0E1B' 0064              .WORD   LP05-AL_DEVTYP
                        0E23' 0066              .WORD   LP06-AL_DEVTYP
                        0E2B' 0068              .WORD   LP07-AL_DEVTYP
                        0E33' 006A              .WORD   LP11-AL_DEVTYP
                        0E64' 006C              .WORD   LP14-AL_DEVTYP
                        0E6C' 006E              .WORD   LP25-AL_DEVTYP
                        0E74' 0070              .WORD   LP26-AL_DEVTYP
                        0E7C' 0072              .WORD   LP27-AL_DEVTYP
                        0E84' 0074              .WORD   LPA11K-AL_DEVTYP
                        0EB7' 0076              .WORD   MA780-AL_DEVTYP
                        0EF3' 0078              .WORD   MBE-AL_DEVTYP
                        0EFA' 007A              .WORD   ML11-AL_DEVTYP
                        0F36' 007C              .WORD   MS750-AL_DEVTYP
                        0F3F' 007E              .WORD   MS780-AL_DEVTYP
                        0F6E' 0080              .WORD   PCL11-AL_DEVTYP
                        0FA0' 0082              .WORD   PX780-AL_DEVTYP
                        0FCF' 0084              .WORD   R80-AL_DEVTYP
                        0FD6' 0086              .WORD   RA60-AL_DEVTYP
                        0FDE' 0088              .WORD   RA80-AL_DEVTYP
                        0FE6' 008A              .WORD   RA81-AL_DEVTYP
                        0FEE' 008C              .WORD   RB730-AL_DEVTYP
                        0FF7' 008E              .WORD   RCF25-AL_DEVTYP
                        1000' 0090              .WORD   RC25-AL_DEVTYP
                        1008' 0092              .WORD   RD51-AL_DEVTYP
                        1010' 0094              .WORD   RD52-AL_DEVTYP
                        1018' 0096              .WORD   RH750-AL_DEVTYP
                        102D' 0098              .WORD   RH780-AL_DEVTYP
                        104E' 009A              .WORD   RL01-AL_DEVTYP
                        1056' 009C              .WORD   RL02-AL_DEVTYP
                        105E' 009E              .WORD   RL11-AL_DEVTYP
                        108F' 00A0              .WORD   RK06-AL_DEVTYP
                        1097' 00A2              .WORD   RK07-AL_DEVTYP
                        109F' 00A4              .WORD   RK611-AL_DEVTYP
```

```
10D1' 00A6              .WORD   RM03-AL_DEVTYP
10D9' 00A8              .WORD   RM05-AL_DEVTYP
10E1' 00AA              .WORD   RM80-AL_DEVTYP
10E9' 00AC              .WORD   RP04-AL_DEVTYP
10F1' 00AE              .WORD   RP05-AL_DEVTYP
10F9' 00B0              .WORD   RP06-AL_DEVTYP
1101' 00B2              .WORD   RP07-AL_DEVTYP
1109' 00B4              .WORD   RQDX1-AL_DEVTYP
111E' 00B6              .WORD   RX02-AL_DEVTYP
1126' 00B8              .WORD   RX211-AL_DEVTYP
1158' 00BA              .WORD   RX50-AL_DEVTYP
1160' 00BC              .WORD   SBIA-AL_DEVTYP
1168' 00BE              .WORD   TE16-AL_DEVTYP
1170' 00C0              .WORD   TM03-AL_DEVTYP
1187' 00C2              .WORD   TM78-AL_DEVTYP
119E' 00C4              .WORD   TS04-AL_DEVTYP
11CF' 00C6              .WORD   TS05-AL_DEVTYP
1218' 00C8              .WORD   TS11-AL_DEVTYP
1249' 00CA              .WORD   TU45-AL_DEVTYP
1251' 00CC              .WORD   TU58-AL_DEVTYP
1259' 00CE              .WORD   TU77-AL_DEVTYP
1261' 00D0              .WORD   TU78-AL_DEVTYP
1269' 00D2              .WORD   TU80-AL_DEVTYP
12B2' 00D4              .WORD   TU81-AL_DEVTYP
12E3' 00D6              .WORD   UBE-AL_DEVTYP
1307' 00D8              .WORD   UDA50-AL_DEVTYP
134F' 00DA              .WORD   UNA11-AL_DEVTYP
1381' 00DC              .WORD   VS100-AL_DEVTYP
13B3' 00DE              .WORD   VS125-AL_DEVTYP
13E5' 00E0              .WORD   VS300-AL_DEVTYP
1417' 00E2              .WORD   VT50-AL_DEVTYP
141F' 00E4              .WORD   VT52-AL_DEVTYP
1427' 00E6              .WORD   VT55-AL_DEVTYP
142F' 00E8              .WORD   VT100-AL_DEVTYP
1438' 00EA              .WORD   VT220-AL_DEVTYP
1441' 00EC              .WORD   VT240-AL_DEVTYP
```

```
                                    00EE   230          .LIST   MEB,MC
                                    00EE   231
                                    00EE   232 AA11K:   $DS_AA11K
                                    00EE               .SAVE   LOCAL_BLOCK
                                    00EE               .PSECT  $ABS$,ABS
                                    0032               .IIF    NB,AA11K$B_BR,  AA11K$B_BR:
                           00000033 0032               .IIF    NB,.BLKB,       .BLKB   1
                                    0033               .IIF    NB,AA11K$K_LEN, AA11K$K_LEN:
                                    00EE               .RESTORE
                  4B 31 31 41 41 00' 00EE               .ASCIC  "AA11K" ; AA11K name
                                    05   00EE
                                    04   00F4               .BYTE   4               ; Maximum unit number
                                    80   00F5               .BYTE   ^X80            ; Constant to identify start of descriptor
                                    83   00F6               .BYTE   ^X83            ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 00F7               .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                                    0B   00F7
                     0003FFFE 0003E000  0103               .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                    83   010B               .BYTE   ^X83            ; Indicate scan octal
                  52 4F 54 43 45 56 00' 010C               .ASCIC  "VECTOR"        ; VECTOR string
                                    06   010C
                     000001FE 00000002  0113               .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                    82   011B               .BYTE   ^X82            ; Indicate scan decimal
                           52 42 00'    011C               .ASCIC  "BR"    ; BR string
                                    02   011C
                     00000007 00000004  011F               .LONG   4,7     ; 4 , 7 limits on input
                                    81   0127               .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0128   233 AD11K:   $DS_AD11K
                                    0128               .SAVE   LOCAL_BLOCK
                                    0128               .PSECT  $ABS$,ABS
                           00000032 0033               .=HP$A_DEPENDENT
                                    0032               .IIF    NB,AD11K$B_BR,  AD11K$B_BR:
                           00000033 0032               .IIF    NB,.BLKB,       .BLKB   1
                                    0033               .IIF    NB,AD11K$K_LEN, AD11K$K_LEN:
                                    00000128           .RESTORE
                  4B 31 31 44 41 00' 0128               .ASCIC  "AD11K" ; AD11K name
                                    05   0128
                                    04   012E               .BYTE   4               ; Maximum unit number
                                    80   012F               .BYTE   ^X80            ; Constant to identify start of descriptor
                                    83   0130               .BYTE   ^X83            ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 0131               .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                                    0B   0131
                     0003FFFE 0003E000  013D               .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                    83   0145               .BYTE   ^X83            ; Indicate scan octal
                  52 4F 54 43 45 56 00' 0146               .ASCIC  "VECTOR"        ; VECTOR string
                                    06   0146
                     000001FE 00000002  014D               .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                    82   0155               .BYTE   ^X82            ; Indicate scan decimal
                           52 42 00'    0156               .ASCIC  "BR"    ; BR string
                                    02   0156
                     00000007 00000004  0159               .LONG   4,7     ; 4 , 7 limits on input
                                    81   0161               .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0162   234 CI_NODE:$DS_CI_NODE                                                    ;[12]
               45 44 4F 4E 5F 49 43 00' 0162               .ASCIC  "CI_NODE"       ; CI_NODE name
                                    07   0162
                                    00   016A               .BYTE   0               ; Maximum unit number
                                    80   016B               .BYTE   ^X80            ; Constant to identify start of descriptor
                                    85   016C               .BYTE   ^X85            ; Indicate scan and verify string
```

```
      65 70 79 74 5F 65 64 6F 4E 00' 016D                .ASCIC  ''Node_type''     ; Node_type string
                                     09  016D
                                     00' 0177            .ASCIC  '"'
                                     00  0177
                                     00' 0178            .ASCIC  '"'
                                     00  0178
               30 38 37 58 41 56 00' 0179                .ASCIC  ''VAX780''
                                     06  0179
               30 35 37 58 41 56 00' 0180                .ASCIC  ''VAX750''
                                     06  0180
                  30 35 43 53 48 00' 0187                .ASCIC  ''HSC50''
                                     05  0187
                                     00' 018D            .ASCIC  '"'
                                     00  018D
                     30 31 4C 4B 00' C18E                .ASCIC  ''KL10''
                                     04  018E
                     54 4E 49 43 00' 0193                .ASCIC  ''CINT''
                                     04  0193
                                     00  0198            .BYTE   0                 ; Null length is end of valid ,,
                                     82  0199            .BYTE   ^X82              ; Indicate scan decimal
73 65 72 64 64 61 5F 65 64 6F 4E 00' 019A                .ASCIC  ''Node_address''  ; Node_address string
                                     73  01A6
                                     0C  019A
                  000000FF 00000000  01A7                .LONG   0,255   ; 0 , 255 limits on input
                                     81  01AF            .BYTE   ^X81              ; Indicate end of PT-descriptor
                                         01B0  235 CI750:  $DS_CI750                                ;                  [07]
               30 35 37 49 43 00' 01B0                   .ASCIC  ''CI750'' ; CI750 name
                                     05  01B0
                                     08  01B6            .BYTE   8                 ; Maximum unit number
                                     80  01B7            .BYTE   ^X80              ; Constant to identify start of descriptor
                                     82  01B8            .BYTE   ^X82              ; Indicate scan decimal
                  74 6F 6C 53 00' 01B9                   .ASCIC  ''Slot''  ; Slot string
                                     04  01B9
               0000000F 0000000A  01BE                   .LONG   10,15   ; 10 , 15 limits on input
                                     82  01C6            .BYTE   ^X82              ; Indicate scan decimal
                        52 42 00' 01C7                   .ASCIC  ''BR''    ; BR string
                                     02  01C7
               00000007 00000004  01CA                   .LONG   4,7     ; 4 , 7 limits on input
                                     82  01D2            .BYTE   ^X82              ; Indicate scan decimal
                  65 64 6F 4E 00' 01D3                   .ASCIC  ''Node''  ; Node string
                                     04  01D3
               000000FF 00000000  01D8                   .LONG   0,255   ; 0 , 255 limits on input
                                     81  01E0            .BYTE   ^X81              ; Indicate end of PT-descriptor
                                         01E1  236 CI780:  $DS_CI780                                ;                  [02]
               30 38 37 49 43 00' 01E1                   .ASCIC  ''CI780'' ; CI780 name
                                     05  01E1
                                     08  01E7            .BYTE   8                 ; Maximum unit number
                                     80  01E8            .BYTE   ^X80              ; Constant to identify start of descriptor
                                     82  01E9            .BYTE   ^X82              ; Indicate scan decimal
                        52 54 00' 01EA                   .ASCIC  ''TR''    ; TR string
                                     02  01EA
               0000000F 00000001  01ED                   .LONG   1,15    ; 1 , 15 limits on input
                                     82  01F5            .BYTE   ^X82              ; Indicate scan decimal
                        52 42 00' 01F6                   .ASCIC  ''BR''    ; BR string
                                     02  01F6
               00000007 00000004  01F9                   .LONG   4,7     ; 4 , 7 limits on input
                                     82  0201            .BYTE   ^X82              ; Indicate scan decimal
```

```
                65 64 6F 4E 00' 0202                .ASCIC  "Node"  ; Node string
                            04  0202
              000000FF 00000000 0207                .LONG   0,255   ; 0 , 255 limits on input
                            81  020F                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                0210    237 CONSOLE:$DS_CONSOLE                                 ;                       [15]
                                0210                .SAVE   LOCAL_BLOCK
                                0210                .PSECT  $ABS$,ABS
                          00000032  0033            .=Hp$A_Dependent
                                0032                .IIF    NB,Console$K_Len,       Console$K_Len:
                            00000210                .RESTORE
          65 6C 6F 73 6E 6F 63 00' 0210             .ASCIC  "console"       ; console name
                            07  0210
                            01  0218                .BYTE   1               ; Maximum unit number
                            80  0219                .BYTE   ^X80            ; Constant to identify start of descriptor
                            81  021A                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                021B    238 CR11:   $DS_CR11
                                021B                .SAVE   LOCAL_BLOCK
                                021B                .PSECT  $ABS$,ABS
                          00000032  0033            .=HP$A_DEPENDENT
                                0032                .IIF    NB,CR11$L_CSR,  CR11$L_CSR:
                          00000036  0032            .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,CR11$B_BR,   CR11$B_BR:
                          00000037  0036            .IIF    NB,.BLKB,       .BLKB   1
                                0037                .IIF    NB,CR11$K_LEN,  CR11$K_LEN:
                            0000021B                .RESTORE
                31 31 52 43 00' 021B                .ASCIC  "CR11"  ; CR11 name
                            04  021B
                            01  0220                .BYTE   1               ; Maximum unit number
                            80  0221                .BYTE   ^X80            ; Constant to identify start of descriptor
                            83  0222                .BYTE   ^X83            ; Indicate scan octal
                     52 53 43 00' 0223              .ASCIC  "CSR"   ; CSR string
                            03  0223
           0003FFFE 0003E000  0227                  .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                            83  022F                .BYTE   ^X83            ; Indicate scan octal
          52 4F 54 43 45 56 00' 0230                .ASCIC  "VECTOR"        ; VECTOR string
                            06  0230
           000001FE 00000002  0237                  .LONG   ^02,^0776       ; 2 , 776 limits on input
                            82  023F                .BYTE   ^X82            ; Indicate scan decimal
                     52 42 00' 0240                 .ASCIC  "BR"    ; BR string
                            02  0240
           00000007 00000004  0243                  .LONG   4,7     ; 4 , 7 limits on input
                            81  024B                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                024C    239 DEQNA:  $DS_DEQNA                                       ;               [19]
                                024C                .SAVE   LOCAL_BLOCK
                                024C                .PSECT  $ABS$,ABS
                          00000032  0037            .=HP$A_DEPENDENT
                                0032                .IIF    NB,DEQNA$L_CSR, DEQNA$L_CSR:
                          00000036  0032            .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,DEQNA$K_LEN, DEQNA$K_LEN:
                            0000024C                .RESTORE
           41 4E 51 45 44 00' 024C                  .ASCIC  "DEQNA" ; DEQNA name
                            05  024C
                            00  0252                .BYTE   0               ; Maximum unit number
                            80  0253                .BYTE   ^X80            ; Constant to identify start of descriptor
                            83  0254                .BYTE   ^X83            ; Indicate scan octal
                     52 53 43 00' 0255              .ASCIC  "CSR"   ; CSR string
                            03  0255
```

```
                0003FFFC 0003E000  0259                  .LONG   ^0760000,^077777         ; 760000 , 777774 limits on input
                              81   0261                  .BYTE   ^X81                     ; Indicate end of PT-descriptor
                                   0262    240 DISK:     $DS_DISK                                                          ;  [18]
                                   0262                  .SAVE   LOCAL_BLOCK
                                   0262                  .PSECT  $ABS$,ABS
                          00000032 0037                  .=HP$A_DEPENDENT
                                   0032                  .IIF    NB,HP$K_DISK_LEN,       HP$K_DISK_LEN:
                                   0032                  .IIF    NB,DISK$K_LEN,  DISK$K_LEN:
                          00000262 0262                  .RESTORE
                   4B 53 49 44 00' 0262                  .ASCIC  "DISK"  ; DISK name
                                04 0262
                                FF 0267                  .BYTE   255                     ; Maximum unit number
                                80 0268                  .BYTE   ^X80                    ; Constant to identify start of descriptor
                                81 0269                  .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                   026A    241 DL11:     $DS_DL11
                                   026A                  .SAVE   LOCAL_BLOCK
                                   026A                  .PSECT  $ABS$,ABS
                          000000?2 0037                  .=HP$A_DEPENDENT
                                   0032                  .IIF    NB,DL11$L_CSR,  DL11$L_CSR:
                          00000036 0032                  .IIF    NB,,BLKL,               .BLKL   1
                                   0036                  .IIF    NB,DL11$B_BR,   DL11$B_BR:
                          00000037 0036                  .IIF    NB,,BLKB,               .BLKB   1
                                   0037                  .IIF    NB,DL11$K_LEN,  DL11$K_LEN:
                          0000026A 026A                  .RESTORE
                   31 31 4C 44 00' 026A                  .ASCIC  "DL11"  ; DL11 name
                                04 026A
                                00 026F                  .BYTE   0                       ; Maximum unit number
                                80 0270                  .BYTE   ^X80                    ; Constant to identify start of descriptor
                                83 0271                  .BYTE   ^X83                    ; Indicate scan octal
                      52 53 43 00' 0272                  .ASCIC  "CSR"   ; CSR string
                                03 0272
                0003FFFE 0003E000  0276                  .LONG   ^0760000,^077777         ; 760000 , 777776 limits on input
                              83   027E                  .BYTE   ^X83                    ; Indicate scan octal
             52 4F 54 43 45 56 00' 027F                  .ASCIC  "VECTOR"                ; VECTOR string
                                06 027F
                000001FE 00000002  0286                  .LONG   ^02,^0776               ; 2 , 776 limits on input
                              82   028E                  .BYTE   ^X82                    ; Indicate scan decimal
                      52 42 00'    028F                  .ASCIC  "BR"    ; BR string
                                02 028F
                00000007 00000004  0292                  .LONG   4,7     ; 4 , 7 limits on input
                              81   029A                  .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                   029B    242 DLVJ1:    $DS_DLVJ1                                                         ;  [19]
                                   029B                  .SAVE   LOCAL_BLOCK
                                   029B                  .PSECT  $ABS$,ABS
                          00000032 0037                  .=HP$A_DEPENDENT
                                   0032                  .IIF    NB,DLVJ1$L_CSR, DLVJ1$L_CSR:
                          0000003o 0032                  .IIF    NB,,BLKL,               .BLKL   1
                                   0036                  .IIF    NB,DLVJ1$B_FLAGS,       DLVJ1$B_FLAGS:
                          00C00037 0036                  .IIF    NB,,BLKB,               .BLKB   1
                                   0037                  .IIF    NB,DLVJ1$K_LEN, DLVJ1$K_LEN:
                          00000298 0298                  .RESTORE
                   31 4A 56 4C 44 00' 029B               .ASCIC  "DLVJ1" ; DLVJ1 name
                                05 029B
                                00 02A1                  .BYTE   0                       ; Maximum unit number
                                80 02A2                  .BYTE   ^X80                    ; Constant to identify start of descriptor
                                83 02A3                  .BYTE   ^X83                    ; Indicate scan octal
                      52 53 43 00' 02A4                  .ASCIC  "CSR"   ; CSR string
```

```
                                        03  02A4
                   0003FFFE 0003E000    02A8                .LONG   ^0760000,^0777776      ; 760000 , 777776 limits on input
                                        83  02B0                .BYTE   ^X83                ; Indicate scan octal
                52 4F 54 43 45 56 00'   02B1                .ASCIC  "VECTOR"            ; VECTOR string
                                        06  02B1
                   000001FE 00000002    02B8                .LONG   ^02,^0776           ; 2 , 776 limits on input
                                        85  02C0                .BYTE   ^X85                ; Indicate scan and verify string
64 20 66 6F 20 72 65 62 6D 75 4E 00'    02C1                .ASCIC  "Number of data bits"   ; Number of data bits string
            73 74 69 62 20 61 74 61     02CD
                                        13  02C1
                            37 00'      02D5                .ASCIC  "7"
                                        01  02D5
                            38 00'      02D7                .ASCIC  "8"
                                        01  02D7
                                        00  C2D9                .BYTE   0                   ; Null length is end of valid 7,8
                                        85  02DA                .BYTE   ^X85                ; Indicate scan and verify string
73 20 66 6F 20 72 65 62 6D 75 4E 00'    02DB                .ASCIC  "Number of stop bits"   ; Number of stop bits string
            73 74 69 62 20 70 6F 74     02E7
                                        13  02DB
                            31 00'      02EF                .ASCIC  "1"
                                        01  02EF
                            32 00'      02F1                .ASCIC  "2"
                                        01  02F1
                                        00  02F3                .BYTE   0                   ; Null length is end of valid 1,2
                                        85  02F4                .BYTE   ^X85                ; For APT, pretend this is a STRING
65 74 65 64 20 79 74 69 72 61 50 00'    02F5                .ASCIC  "Parity detection enabled"
65 6C 62 61 6E 65 20 6E 6F 69 74 63     0301
                                        64  030D
                                        18  02F5
                         4F 4E 00'      030E                .ASCIC  "NO"
                                        02  030E
                      53 45 59 00'      0311                .ASCIC  "YES"
                                        03  0311
                                        00  0315                .BYTE   0
                                        85  0316                .BYTE   ^X85                ; For APT, pretend this is a STRING
79 74 69 72 61 70 20 6E 65 76 45 00'    0317                .ASCIC  "Even parity enabled"
            64 65 6C 62 61 6E 65 20     0323
                                        13  0317
                         4F 4E 00'      0328                .ASCIC  "NO"
                                        02  0328
                      53 45 59 00'      032E                .ASCIC  "YES"
                                        03  032E
                                        00  0332                .BYTE   0
                                        81  0333                .BYTE   ^X81                ; Indicate end of PT-descriptor
                                            0334    243 DMC11:  $DS_DMC11
                                            0334                .SAVE   LOCAL_BLOCK
                                            0334                .PSECT  $ABS$,ABS
                            00000032        0037                .=HP$A_DEPENDENT
                                            0032                .IIF    NB,DMC11$L_CSR, DMC11$L_CSR:
                            00000036        0032                .IIF    NB,.BLKL,       .BLKL   1
                                            0036                .IIF    NB,DMC11$B_BR,  DMC11$B_BR:
                            00000037        0036                .IIF    NB,.BLKB,       .BLKB   1
                                            0037                .IIF    NB,DMC11$K_LEN, DMC11$K_LEN:
                                        00000334                .RESTORE
                31 31 43 4D 44 00'      0334                .ASCIC  "DMC11" ; DMC11 name
                                        05  0334
                                        00  033A                .BYTE   0                   ; Maximum unit number
```

```
                          80  033B                .BYTE   ^X80             ; Constant to identify start of descriptor
                          83  033C                .BYTE   ^X83             ; Indicate scan octal
                52 53 43 00'  033D                .ASCIC  "CSR"   ; CSR string
                          03  033D
            0003FFFE 0003E000  0341                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          83  0349                .BYTE   ^X83             ; Indicate scan octal
          52 4F 54 43 45 56 00'  034A              .ASCIC  "VECTOR"        ; VECTOR string
                          06  034A
            000001FE 00000002  0351                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          82  0359                .BYTE   ^X82             ; Indicate scan decimal
                    52 42 00'  035A                .ASCIC  "BR"    ; BR string
                          02  035A
            00000007 00000004  035D                .LONG   4,7     ; 4 , 7 limits on input
                          81  0365                .BYTE   ^X81             ; Indicate end of PT-descriptor
                              0366    244 DMF32:  $DS_DMF32                               ;                                   [02]
                              0366                .SAVE   LOCAL_BLOCK
                              0366                .PSECT  $ABS$,ABS
                    00000032  0037                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$B_DMF32_COMMON,   HP$B_DMF32_COMMON:
                              0032                .IIF    NB,DMF32$B_COMMON,      DMF32$B_COMMON:
                    00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                              0033                .IIF    NB,HP$B_DMF32_FLAGS,    HP$B_DMF32_FLAGS:
                              0033                .IIF    NB,DMF32$B_FLAGS,       DMF32$B_FLAGS:
                    00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                              0034                .IIF    NB,HP$B_DMF32_BR,       HP$B_DMF32_BR:
                              0034                .IIF    NB,DMF32$B_BR,  DMF32$B_BR:
                    00000035  0034                .IIF    NB,.BLKB,       .BLKB   1
                              0035                .IIF    NB,HP$L_DMF32_CSR,      HP$L_DMF32_CSR:
                              0035                .IIF    NB,DMF32$L_CSR, DMF32$L_CSR:
                    00000039  0035                .IIF    NB,.BLKL,       .BLKL   1
                              0039                .IIF    NB,HP$K_DMF32_LEN,      HP$K_DMF32_LEN:
                              0039                .IIF    NB,DMF32$K_LEN, DMF32$K_LEN:
                        00000366                  .RESTORE
          32 33 46 4D 44 00'  0366                .ASCIC  "DMF32" ; DMF32 name
                          05  0366
                          05  036C                .BYTE   5                ; Maximum unit number
                          80  036D                .BYTE   ^X80             ; Constant to identify start of descriptor
                          83  036E                .BYTE   ^X83             ; Indicate scan octal
                52 53 43 00'  036F                .ASCIC  "CSR"   ; CSR string
                          03  036F
            0003FFFE 0003E000  0373                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          83  037B                .BYTE   ^X83             ; Indicate scan octal
          72 6F 74 63 65 56 00'  037C              .ASCIC  "Vector"        ; Vector string
                          06  037C
            000001FE 00000000  0383                .LONG   ^O300,^O776     ; 300 , 776 limits on input
                          82  038B                .BYTE   ^X82             ; Indicate scan decimal
                    52 42 00'  038C                .ASCIC  "BR"    ; BR string
                          02  038C
            00000007 00000004  038F                .LONG   4,7     ; 4 , 7 limits on input
                          81  0397                .BYTE   ^X81             ; Indicate end of PT-descriptor
                              0398    245 DMF32A: $DS_DMF32A                              ;                                   [02]
                              0398                .SAVE   LOCAL_BLOCK
                              0398                .PSECT  $ABS$,ABS
                    00000032  0039                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$B_DMF32A_BR,      HP$B_DMF32A_BR:
                              0032                .IIF    NB,DMF32A$B_BR, DMF32A$B_BR:
                    00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
```

```
                          0033                .IIF    NB,HP$B_DMF32A_ACTIV,   HP$B_DMF32A_ACTIV:
                          0033                .IIF    NB,DMF32A$B_ACTIV,      DMF32A$B_ACTIV:
                 00000034 0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,HP$B_DMF32A_SPEED,   HP$B_DMF32A_SPEED:
                          0034                .IIF    NB,DMF32A$B_SPEED,      DMF32A$B_SPEED:
                 00000035 0034                .IIF    NB,.BLKB,       .BLKB   1
                          0035                .IIF    NB,HP$W_DMF32A_FLAGS,   HP$W_DMF32A_FLAGS:
                          0035                .IIF    NB,DMF32A$W_FLAGS,      DMF32A$W_FLAGS:
                 00000037 0035                .IIF    NB,.BLKW,       .BLKW   1
                          0037                .IIF    NB,HP$B_DMF32A_JUMP,    HP$B_DMF32A_JUMP:
                          0037                .IIF    NB,DMF32A$B_JUMP,       DMF32A$B_JUMP:
                 00000038 0037                .IIF    NB,.BLKB,       .BLKB   1
                          0038                .IIF    NB,HP$K_DMF32A_LEN,     HP$K_DMF32A_LEN:
                          0038                .IIF    NB,DMF32A$K_LEN,        DMF32A$K_LEN:
                      00000398                .RESTORE
        41 32 33 46 4D 44 00' 0398            .ASCIC  "DMF32A"        ; DMF32A name
                          06  0398
                          00  039F            .BYTE   0               ; Maximum unit number
                          80  03A0            .BYTE   ^X80            ; Constant to identify start of descriptor
                          83  03A1            .BYTE   ^X83            ; Indicate scan octal
                  52 53 43 00' 03A2           .ASCIC  "CSR"   ; CSR string
                          03  03A2
     0003FFFE 0003E000    03A6                .LONG   ^00760000,^00777776     ; 0760000 , 0777776 limits on input
                          83  03AE            .BYTE   ^X83            ; Indicate scan octal
  72 6F 74 63 65 56 00' 03AF                .ASCIC  "Vector"        ; Vector string
                          06  03AF
        000001FE 000000C0    03B6           .LONG   ^0300,^0776     ; 300 , 776 limits on input
                          82  03BE            .BYTE   ^X82            ; Indicate scan decimal
                  52 42 00' 03BF            .ASCIC  "BR"    ; BR string
                          02  03BF
        00000007 00000004    03C2           .LONG   4,7     ; 4 , 7 limits on input
                          83  03CA            .BYTE   ^X83            ; Indicate scan octal
65 6E 69 4C 20 65 76 69 74 63 41 00' 03CB    .ASCIC  "Active Lines"  ; Active Lines string
                          73  03D7
                          0C  03CB
        000000FF 00000001    03D8           .LONG   ^01,^0377       ; 1 , 377 limits on input
                          85  03E0            .BYTE   ^X85            ; Indicate scan and verify string
     65 74 61 52 20 64 75 61 42 00' 03E1     .ASCIC  "Baud Rate"     ; Baud Rate string
                          09  03E1
                  30 35 00' 03EB            .ASCIC  "50"
                          02  03EB
                  35 37 00' 03EE            .ASCIC  "75"
                          02  03EE
               30 31 31 00' 03F1            .ASCIC  "110"
                          03  03F1
            35 2E 34 33 31 00' 03F5         .ASCIC  "134.5"
                          05  03F5
               30 35 31 00' 03FB            .ASCIC  "150"
                          03  03FB
               30 30 33 00' 03FF            .ASCIC  "300"
                          03  03FF
               30 30 36 00' 0403            .ASCIC  "600"
                          03  0403
            30 30 32 31 00' 0407            .ASCIC  "1200"
                          04  0407
            30 30 38 31 00' 040C            .ASCIC  "1800"
                          04  040C
```

```
                     30 30 30 32 00' 0411              .ASCIC  "2000"
                                 04  0411
                     30 30 34 32 00' 0416              .ASCIC  "2400"
                                 04  0416
                     30 30 36 33 00' 041B              .ASCIC  "3600"
                                 04  041B
                     30 30 38 34 00' 0420              .ASCIC  "4800"
                                 04  0420
                     30 30 32 37 00' 0425              .ASCIC  "7200"
                                 04  0425
                     30 30 36 39 00' 042A              .ASCIC  "9600"
                                 04  042A
                  30 30 32 39 31 00' 042F              .ASCIC  "19200"
                                 05  042F
                                 00  0435              .BYTE   0               ; Null length is end of valid 50,75,110,134.5,150,30
                                 85  0436              .BYTE   ^X85            ; Indicate scan and verify string
79 54 20 6B 63 61 62 70 6F 6F 4C 00' 0437              .ASCIC  "Loopback Type" ; Loopback Type string
                              65 70  0443
                                 0D  0437
            4C 41 4E 52 45 54 4E 49 00' 0445              .ASCIC  "INTERNAL"
                                 08  0445
                     38 34 32 33 48 00' 044E              .ASCIC  "H3248"
                                 05  044E
4D 45 44 4F 4D 5F 4C 41 43 4F 4C 00' 0454              .ASCIC  "LOCAL_MODEM"
                                 0B  0454
4C 42 41 4D 4D 41 52 47 4F 52 50 00' 0460              .ASCIC  "PROGRAMMABLE_MODEM"
                  4D 45 44 4F 4D 5F 45  046C
                                 12  0460
                     39 34 32 33 48 00' 0473              .ASCIC  "H3249"
                                 05  0473
                                 00  0479              .BYTE   0               ; Null length is end of valid INTERNAL,<H3248>,LOCAL
                                 85  047A              .BYTE   ^X85            ; For APT, pretend this is a STRING
74 69 6E 49 20 73 75 62 69 6E 55 00' 047B              .ASCIC  "Unibus Init Jumper"
                  72 65 70 6D 75 4A 20  0487
                                 12  047B
                        4F 4E 00' 048E              .ASCIC  "NO"
                                 02  048E
                     53 45 59 00' 0491              .ASCIC  "YES"
                                 03  0491
                                 00  0495              .BYTE   0
                                 81  0496              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                     0497  246 DMF32P: $DS_DMF32P                                ;                                [02]
                                     0497              .SAVE   LOCAL_BLOCK
                                     0497              .PSECT  $ABS$,ABS
                           00000032  0039              .=HP$A_DEPENDENT
                                     0032              .IIF    NB,HP$B_DMF32P_COMMON,  HP$B_DMF32P_COMMON:
                                     0032              .IIF    NB,DMF32P$B_COMMON,     DMF32P$B_COMMON:
                           00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                                     0033              .IIF    NB,HP$B_DMF32P_FLAGS,   HP$B_DMF32P_FLAGS:
                                     0033              .IIF    NB,DMF32P$B_FLAGS,      DMF32P$B_FLAGS:
                           00000034  0033              .IIF    NB,.BLKB,       .BLKB   1
                                     0034              .IIF    NB,HP$B_DMF32P_BR,      HP$B_DMF32P_BR:
                                     0034              .IIF    NB,DMF32P$B_BR, DMF32P$B_BR:
                           00000035  0034              .IIF    NB,.BLKB,       .BLKB   1
                                     0035              .IIF    NB,HP$L_DMF32P_CSR,     HP$L_DMF32P_CSR:
                                     0035              .IIF    NB,DMF32P$L_CSR,        DMF32P$L_CSR:
                           00000039  0035              .IIF    NB,.BLKL,       .BLKL   1
```

```
                                0039                    .IIF    NB,HP$K_DMF32P_LEN,     HP$K_DMF32P_LEN:
                                0039                    .IIF    NB,DMF32P$K_LEN,        DMF32P$K_LEN:
                           00000497                     .RESTORE
                  50 32 33 46 4D 44 00' 0497            .ASCIC  "DMF32P"        ; DMF32P name
                                06  0497
                                00  049E                .BYTE   0               ; Maximum unit number
                                80  049F                .BYTE   ^X80            ; Constant to identify start of descriptor
                                83  04A0                .BYTE   ^X83            ; Indicate scan octal
                        52 53 43 00' 04A1               .ASCIC  "CSR"   ; CSR string
                                03  04A1
                     0003FFFE 0003E000  04A5            .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                83  04AD                .BYTE   ^X83            ; Indicate scan octal
                  72 6F 74 63 65 56 00' 04AE            .ASCIC  "Vector"        ; Vector string
                                06  04AE
                     000001FE 000000C0  C4B5            .LONG   ^O300,^O776     ; 300 , 776 limits on input
                                82  04BD                .BYTE   ^X82            ; Indicate scan decimal
                        52 42 00' 04BE                  .ASCIC  "BR"    ; BR string
                                02  04BE
                     00000007 00000004  04C1            .LONG   4,7     ; 4 , 7 limits on input
                                85  04C9                .BYTE   ^X85            ; Indicate scan and verify string
65 63 69 76 65 44 20 74 72 6F 50 00' 04CA               .ASCIC  "Port Device"   ; Port Device string
                                0B  04CA
                  52 45 48 54 4F 00' 04D6               .ASCIC  "OTHER"
                                05  04D6
                        50 4C 00' 04DC                  .ASCIC  "LP"
                                02  04DC
                     50 41 52 57 00' 04DF               .ASCIC  "WRAP"
                                04  04DF
                                00  04E4                .BYTE   0               ; Null length is end of valid OTHER,LP,WRAP
                                81  04E5                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    04E6    247 DMF32S: $DS_DMF32S                              ;                       [02]
                                    04E6                .SAVE   LOCAL_BLOCK
                                    04E6                .PSECT  $ABS$,ABS
                           00000032 0039                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$B_DMF32S_COMMON,  HP$B_DMF32S_COMMON:
                                    0032                .IIF    NB,DMF32S$B_COMMON,     DMF32S$B_COMMON:
                           00000033 0032                .IIF    NB,,.BLKB,      .BLKB   1
                                    0033                .IIF    NB,HP$B_DMF32S_FLAGS,   HP$B_DMF32S_FLAGS:
                                    0033                .IIF    NB,DMF32S$B_FLAGS,      DMF32S$B_FLAGS:
                           00000034 0033                .IIF    NB,,.BLKB,      .BLKB   1
                                    0034                .IIF    NB,HP$B_DMF32S_BR,      HP$B_DMF32S_BR:
                                    0034                .IIF    NB,DMF32S$B_BR, DMF32S$B_BR:
                           00000035 0034                .IIF    NB,,.BLKB,      .BLKB   1
                                    0035                .IIF    NB,HP$L_DMF32S_CSR,     HP$L_DMF32S_CSR:
                                    0035                .IIF    NB,DMF32S$L_CSR,        DMF32S$L_CSR:
                           00000039 0035                .IIF    NB,,.BLKL,      .BLKL   1
                                    0039                .IIF    NB,HP$K_DMF32S_LEN,     HP$K_DMF32S_LEN:
                                    0039                .IIF    NB,DMF32S$K_LEN,        DMF32S$K_LEN:
                                000004E6                .RESTORE
            53 32 33 46 4D 44 00' 04E6                  .ASCIC  "DMF32S"        ; DMF32S name
                                06  04E6
                                01  04ED                .BYTE   1               ; Maximum unit number
                                80  04EE                .BYTE   ^X80            ; Constant to identify start of descriptor
                                83  04EF                .BYTE   ^X83            ; Indicate scan octal
                        52 53 43 00' 04F0               .ASCIC  "CSR"   ; CSR string
                                03  04F0
                     0003FFFE 0003E000  04F4            .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
```

```
                                 83  04FC                .BYTE   ^X83              ; Indicate scan octal
                72 6F 74 63 65 56 00' 04FD               .ASCIC  "Vector"         ; Vector string
                                 06  04FD
                   000001FE 00000002  0504               .LONG   ^02,^0776        ; 2 , 776 limits on input
                                 82  050C                .BYTE   ^X82             ; Indicate scan decimal
                            52 42 00' 050D               .ASCIC  "BR"    ; BR string
                                 02  050D
                   00000007 00000004  0510               .LONG   4,7      ; 4 , 7 limits on input
                                 85  0518                .BYTE   ^X85             ; Indicate scan and verify string
72 57 20 6C 61 6E 72 65 74 78 45 00'  0519               .ASCIC  "External Wrap" ; External Wrap string
                              70 61  0525
                                 0D  0519
                      45 4E 4F 4E 00' 0527               .ASCIC  "NONE"
                                 04  0527
                   4D 45 44 4F 4D 00' C52C               .ASCIC  "MODEM"
                                 05  052C
                      50 41 52 57 00' 0532               .ASCIC  "WRAP"
                                 04  0532
                                 00  0537                .BYTE   0                ; Null length is end of valid NONE,MODEM,WRAP
                                 81  0538                .BYTE   ^X81             ; Indicate end of PT-descriptor
                                     0539    248 DMP11:  $DS_DMP11
                                     0539                .SAVE   LOCAL_BLOCK
                                     0539                .PSECT  $ABS$,ABS
                            00000032 0039                .=HP$A_DEPENDENT
                                     0032                .IIF    NB,DMP11$B_COMMON,       DMP11$B_COMMON:
                            00000033 0032                .IIF    NB,.BLKB,        .BLKB   1
                                     0033                .IIF    NB,DMP11$B_FLAGS,        DMP11$B_FLAGS:
                            00000034 0033                .IIF    NB,.BLKB,        .BLKB   1
                                     0034                .IIF    NB,DMP11$B_BR,   DMP11$B_BR:
                            00000035 0034                .IIF    NB,.BLKB,        .BLKB   1
                                     0035                .IIF    NB,DMP11$L_CSR,  DMP11$L_CSR:
                            00000039 0035                .IIF    NB,.BLKL,        .BLKL   1
                                     0039                .IIF    NB,DMP11$K_LEN,  DMP11$K_LEN:
                                 00000539                .RESTORE
                      31 31 50 4D 44 00' 0539            .ASCIC  "DMP11" ; DMP11 name
                                 05  0539
                                 00  053F                .BYTE   0                ; Maximum unit number
                                 80  0540                .BYTE   ^X80             ; Constant to identify start of descriptor
                                 83  0541                .BYTE   ^X83             ; Indicate scan octal
                         52 53 43 00' 0542               .ASCIC  "CSR"   ; CSR string
                                 03  0542
                   0003FFFE 0003E000  0546               .LONG   ^0760000,^0777776        ; 760000 , 777776 limits on input
                                 83  054E                .BYTE   ^X83             ; Indicate scan octal
                   52 4F 54 43 45 56 00' 054F            .ASCIC  "VECTOR"         ; VECTOR string
                                 06  054F
                   000001FE 00000002  0556               .LONG   ^02,^0776        ; 2 , 776 limits on input
                                 82  055E                .BYTE   ^X82             ; Indicate scan decimal
                            52 42 00' 055F               .ASCIC  "BR"    ; BR string
                                 02  055F
                   00000007 00000004  0562               .LONG   4,7      ; 4 , 7 limits on input
                                 85  056A                .BYTE   ^X85             ; Indicate scan and verify string
61 74 53 20 6C 6F 72 74 6E 6F 43 00'  056B               .ASCIC  "Control Station"        ; Control Station string
                           6E 6F 69 74  0577
                                 0F  056B
                            4F 4E 00' 057B               .ASCIC  "NO"
                                 02  057B
                         53 45 59 00' 057E               .ASCIC  "YES"
```

```
                                03  057E
                                00  0582                .BYTE   0               ; Null length is end of valid NO,YES
                                85  0583                .BYTE   ^X85            ; Indicate scan and verify string
      79 72 61 74 75 62 69 72 54 00' 0584                .ASCIC  "Tributary"     ; Tributary string
                                09  0584
                           4F 4E 00' 058E                .ASCIC  "NO"
                                02  058E
                        53 45 59 00' 0591                .ASCIC  "YES"
                                03  0591
                                00  0595                .BYTE   0               ; Null length is end of valid NO,YES
                                85  0596                .BYTE   ^X85            ; Indicate scan and verify string
74 6C 69 62 61 74 61 70 6D 6F 43 00' 0597                .ASCIC  "Compatabilty Mode"     ; Compatabilty Mode string
                     65 64 6F 4D 20 79  05A3
                                11  0597
                           4F 4E 00' 05A9                .ASCIC  "NO"
                                02  05A9
                        53 45 59 00' 05AC                .ASCIC  "YES"
                                03  05AC
                                00  05B0                .BYTE   0               ; Null length is end of valid NO,YES
                                85  05B1                .BYTE   ^X85            ; Indicate scan and verify string
78 65 6C 70 75 44 20 66 6C 61 48 00' 05B2                .ASCIC  "Half Duplex"   ; Half Duplex string
                                08  05B2
                           4F 4E 00' 05BE                .ASCIC  "NO"
                                02  05BE
                        53 45 59 00' 05C1                .ASCIC  "YES"
                                03  05C1
                                00  05C5                .BYTE   0               ; Null length is end of valid NO,YES
                                81  05C6                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    05C7    249 DMR11:  $DS_DMR11
                                    05C7                .SAVE   LOCAL_BLOCK
                                    05C7                .PSECT  $ABS$,ABS
                          00000032  0039                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,DMR11$L_CSR, DMR11$L_CSR:
                          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                    0036                .IIF    NB,DMR11$B_BR,  DMR11$B_BR:
                          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                    0037                .IIF    NB,DMR11$K_LEN, DMR11$K_LEN:
                                00000  05C7              .RESTORE
   31 31 52 4D 44 00'               05C7                .ASCIC  "DMR11" ; DMR11 name
                                05  05C7
                                00  05CD                .BYTE   0               ; Maximum unit number
                                80  05CE                .BYTE   ^X80            ; Constant to identify start of descriptor
                                83  05CF                .BYTE   ^X83            ; Indicate scan octal
         52 53 43 00'           05D0                .ASCIC  "CSR"   ; CSR string
                                03  05D0
   0003FFFE 0003E000            05D4                .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                                83  05DC                .BYTE   ^X83            ; Indicate scan octal
52 4F 54 43 45 56 00'           05DD                .ASCIC  "VECTOR"        ; VECTOR string
                                06  05DD
   000001FE 00000002            05E4                .LONG   ^02,^0776       ; 2 , 776 limits on input
                                82  05EC                .BYTE   ^X82            ; Indicate scan decimal
            52 42 00'           05ED                .ASCIC  "BR"    ; BR string
                                02  05ED
   00000007 00000004            05F0                .LONG   4,7     ; 4 , 7 limits on input
                                81  05F8                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    05F9    250 DMZ32:  $DS_DMZ32                               ;                       [21]
                                    05F9                .SAVE   LOCAL_BLOCK
```

```
                        05F9            .PSECT  $ABS$,ABS
                    00000032    0039    .=HP$A_DEPENDENT
                                0032    .IIF    NB,HP$B_DMZ32_BR,       HP$B_DMZ32_BR:
                    00000033    0032    .IIF    NB,.BLKB,       .BLKB   1
                                0033    .IIF    NB,HP$B_DMZ32_OCTET0,   HP$B_DMZ32_OCTET0:
                    00000034    0033    .IIF    NB,.BLKB,       .BLKB   1
                                0034    .IIF    NB,HP$B_DMZ32_OCTET1,   HP$B_DMZ32_OCTET1:
                    00000035    0034    .IIF    NB,.BLKB,       .BLKB   1
                                0035    .IIF    NB,HP$B_DMZ32_OCTET2,   HP$B_DMZ32_OCTET2:
                    00000036    0035    .IIF    NB,.BLKB,       .BLKB   1
                                0036    .IIF    NB,HP$B_DMZ32_SPEED,    HP$B_DMZ32_SPEED:
                    00000037    0036    .IIF    NB,.BLKB,       .BLKB   1
                                0037    .IIF    NB,HP$W_DMZ32_LOOP,     HP$W_DMZ32_LOOP:
                    00000039    0037    .IIF    NB,.BLKW,       .BLKW   1
                                C039    .IIF    NB,HP$B_DMZ32_MODEM,    HP$B_DMZ32_MODEM:
                    0000003A    0039    .IIF    NB,.BLKB,       .BLKB   1
                                003A    .IIF    NB,HP$K_DMZ32_LEN,      HP$K_DMZ32_LEN:
                           000005F9     .RESTORE
      32 33 5A 4D 44 00'        05F9    .ASCIC  "DMZ32" ; DMZ32 name
                                05  05F9
                                00  05FF    .BYTE   0               ; Maximum unit number
                                80  0600    .BYTE   ^X80            ; Constant to identify start of descriptor
                                83  0601    .BYTE   ^X83            ; Indicate scan octal
            52 53 43 00'        0602    .ASCIC  "CSR"   ; CSR string
                                03  0602
      0003FFFE 0003E000         0606    .LONG   ^O760000,^O00777776     ; 760000 , 0777776 limits on input
                                83  060E    .BYTE   ^X83            ; Indicate scan octal
   72 6F 74 63 65 56 00'        060F    .ASCIC  "Vector"        ; Vector string
                                06  060F
      000001FE 000000C0         0616    .LONG   ^O300,^O776     ; 300 , 776 limits on input
                                82  061E    .BYTE   ^X82            ; Indicate scan decimal
                        52 42 00'   061F    .ASCIC  "BR"    ; BR string
                                02  061F
                    00000006 00000005   0622    .LONG   5,6     ; 5 , 6 limits on input
                                83  062A    .BYTE   ^X83            ; Indicate scan octal
20 53 45 4E 49 4C 20 54 53 45 54 00'    062B    .ASCIC  "TEST LINES (OCTET0)"   ; TEST LINES (OCTET0) string
            29 30 54 45 54 43 4F 28     0637
                                13  062B
                    000000FF 00000000   063F    .LONG   ^O0,^O0377      ; 0 , 377 limits on input
                                83  0647    .BYTE   ^X83            ; Indicate scan octal
4C 4E 4F 20 33 20 4C 45 56 45 4C 00'    0648    .ASCIC  "LEVEL 3 ONLY - TEST LINES (OCTET1)"    ; LEVEL 3 ONLY - TEST LINES
4E 49 4C 20 54 53 45 54 20 2D 20 59     0654
   29 31 54 45 54 43 4F 28 20 53 45     0660
                                22  0648
                    000000FF 00000000   066B    .LONG   ^O0,^O0377      ; 0 , 377 limits on input
                                83  0673    .BYTE   ^X83            ; Indicate scan octal
4C 4E 4F 20 33 20 4C 45 56 45 4C 00'    0674    .ASCIC  "LEVEL 3 ONLY - TEST LINES (OCTET2)"    ; LEVEL 3 ONLY - TEST LINES
4E 49 4C 20 54 53 45 54 20 2D 20 59     0680
   29 32 54 45 54 43 4F 28 20 53 45     068C
                                22  0674
                    000000FF 00000000   0697    .LONG   ^O0,^O0377      ; 0 , 377 limits on input
                                85  069F    .BYTE   ^X85            ; Indicate scan and verify string
      65 74 61 52 20 64 75 61 42 00'    06A0    .ASCIC  "Baud Rate"     ; Baud Rate string
                                09  06A0
                           30 35 00'    06AA    .ASCIC  "50"
                                02  06AA
                           35 37 00'    06AD    .ASCIC  "75"
```

```
                                  02  06AD
                           30 31 31 00' 06B0              .ASCIC  "110"
                                  03  06B0
                        35 2E 34 33 31 00' 06B4           .ASCIC  "134.5"
                                  05  06B4
                           30 35 31 00' 06BA              .ASCIC  "150"
                                  03  06BA
                           30 30 33 00' 06BE              .ASCIC  "300"
                                  03  06BE
                           30 30 36 00' 06C2              .ASCIC  "600"
                                  03  06C2
                        30 30 32 31 00' 06C6              .ASCIC  "1200"
                                  04  06C6
                        30 30 38 31 00' 06CB              .ASCIC  "1800"
                                  04  C6CB
                        30 30 30 32 00' 06D0              .ASCIC  "2000"
                                  04  06D0
                        30 30 34 32 00' 06D5              .ASCIC  "2400"
                                  04  06D5
                        30 30 38 34 00' 06DA              .ASCIC  "4800"
                                  04  06DA
                        30 30 36 39 00' 06DF              .ASCIC  "9600"
                                  04  06DF
                     30 30 32 39 31 00' 06E4              .ASCIC  "19200"
                                  05  06E4
                                  00  06EA              .BYTE   0               ; Null length is end of valid 50,75,110,134.5,150,30
                                  85  06EB              .BYTE   ^X85            ; Indicate scan and verify string
79 54 20 6B 63 61 62 70 6F 6F 4C 00' 06EC              .ASCIC  "Loopback Type" ; Loopback Type string
                              65 70  06F8
                                  0D  06EC
                                  30 00' 06FA          .ASCIC  "0"
                                  01  06FA
                                  31 00' 06FC          .ASCIC  "1"
                                  01  06FC
                                  32 00' 06FE          .ASCIC  "2"
                                  01  06FE
                                  33 00' 0700          .ASCIC  "3"
                                  01  0700
                                  34 00' 0702          .ASCIC  "4"
                                  01  0702
                                  35 00' 0704          .ASCIC  "5"
                                  01  0704
                                  36 00' 0706          .ASCIC  "6"
                                  01  0706
                                  37 00' 0708          .ASCIC  "7"
                                  01  0708
                                  38 00' 070A          .ASCIC  "8"
                                  01  070A
                                  00  070C              .BYTE   0               ; Null length is end of valid 0,1,2,3,4,5,6,7,8
                                  85  070D              .BYTE   ^X85            ; For APT, pretend this is a STRING
72 74 6E 6F 43 20 6D 65 64 6F 4D 00' 070E              .ASCIC  "Modem Control"
                              6C 6F  071A
                                  0D  070E
                              4F 4E 00' 071C           .ASCIC  "NO"
                                  02  071C
                           53 45 59 00' 071F           .ASCIC  "YES"
                                  03  071F
```

```
                        00  0723                .BYTE   0
                        81  0724                .BYTE   ^X81             ; Indicate end of PT-descriptor
                            0725    251 DR11B:  $DS_DR11B
                            0725                .SAVE   LOCAL_BLOCK
                            0725                .PSECT  $ABS$,ABS
                  00000032  003A                .=HP$A_DEPENDENT
                            0032                .IIF    NB,DR11B$L_CSR, DR11B$L_CSR:
                  00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                            0036                .IIF    NB,DR11B$B_BR,  DR11B$B_BR:
                  00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                            0037                .IIF    NB,DR11B$K_LEN, DR11B$K_LEN:
                  00000725  0725                .RESTORE
            42 31 31 52 44 00'  0725            .ASCIC  "DR11B" ; DR11B name
                        05  0725
                        00  072B                .BYTE   0                ; Maximum unit number
                        80  072C                .BYTE   ^X80             ; Constant to identify start of descriptor
                        83  072D                .BYTE   ^X83             ; Indicate scan octal
               52 53 43 00'  072E               .ASCIC  "CSR"   ; CSR string
                        03  072E
            0003FFFE 0003E000  0732             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        83  073A                .BYTE   ^X83             ; Indicate scan octal
         52 4F 54 43 45 56 00'  073B            .ASCIC  "VECTOR"         ; VECTOR string
                        06  073B
            000001FE 00000002  0742             .LONG   ^O2,^O776        ; 2 , 776 limits on input
                        82  074A                .BYTE   ^X82             ; Indicate scan decimal
                  52 42 00'  074B               .ASCIC  "BR"    ; BR string
                        02  074B
            00000007 00000004  074E             .LONG   4,7     ; 4 , 7 limits on input
                        81  0756                .BYTE   ^X81             ; Indicate end of PT-descriptor
                            0757    252 DR11K:  $DS_DR11K
                            0757                .SAVE   LOCAL_BLOCK
                            0757                .PSECT  $ABS$,ABS
                  00000032  003A                .=HP$A_DEPENDENT
                            0032                .IIF    NB,DR11K$B_BR,  DR11K$B_BR:
                  00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                            0033                .IIF    NB,DR11K$K_LEN, DR11K$K_LEN:
                  00000757  0757                .RESTORE
            4B 31 31 52 44 00'  0757            .ASCIC  "DR11K" ; DR11K name
                        05  0757
                        06  075D                .BYTE   6                ; Maximum unit number
                        80  075E                .BYTE   ^X80             ; Constant to identify start of descriptor
                        83  075F                .BYTE   ^X83             ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00'  0760      .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                        0B  0760
            0003FFFE 0003E000  076C             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        83  0774                .BYTE   ^X83             ; Indicate scan octal
         52 4F 54 43 45 56 00'  0775            .ASCIC  "VECTOR"         ; VECTOR string
                        06  0775
            000001FE 00000002  077C             .LONG   ^O2,^O776        ; 2 , 776 limits on input
                        82  0784                .BYTE   ^X82             ; Indicate scan decimal
                  52 42 00'  0785               .ASCIC  "BR"    ; BR string
                        02  0785
            00000007 00000004  0788             .LONG   4,7     ; 4 , 7 limits on input
                        81  0790                .BYTE   ^X81             ; Indicate end of PT-descriptor
                            0791    253 DR11W:  $DS_DR11W
                            0791                .SAVE   LOCAL_BLOCK
                            0791                .PSECT  $ABS$,ABS
```

```
                    00000032  003A                .=HP$A_DEPENDENT
                              0032                .IIF    NB,DR11W$L_CSR, DR11W$L_CSR:
                    00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                              0036                .IIF    NB,DR11W$B_BR,  DR11W$B_BR:
                    00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                              0037                .IIF    NB,DR11W$K_LEN, DR11W$K_LEN:
                          00000791                .RESTORE
              57 31 31 52 44 00'  0791            .ASCIC  "DR11W" ; DR11W name
                              05  0791
                              00  0797            .BYTE   0               ; Maximum unit number
                              80  0798            .BYTE   ^X80            ; Constant to identify start of descriptor
                              83  0799            .BYTE   ^X83            ; Indicate scan octal
                    52 53 43 00'  079A            .ASCIC  "CSR"   ; CSR string
                              03  079A
              0003FFFE 0003E000   C79E            .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                              83  07A6            .BYTE   ^X83            ; Indicate scan octal
           52 4F 54 43 45 56 00'  07A7            .ASCIC  "VECTOR"        ; VECTOR string
                              06  07A7
              000001FE 00000002   07AE            .LONG   ^O2,^O776       ; 2 , 776 limits on input
                              82  07B6            .BYTE   ^X82            ; Indicate scan decimal
                    52 42 00'     07B7            .ASCIC  "BR"    ; BR string
                              02  07B7
              00000007 00000004   07BA            .LONG   4,7     ; 4 , 7 limits on input
                              81  07C2            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                  07C3    254 DR750:  $DS_DR750
                                  07C3            .SAVE   LOCAL_BLOCK
                                  07C3            .PSECT  $ABS$,ABS
                    00000032      003A            .=HP$A_DEPENDENT
                              0032                .IIF    NB,DR750$B_SLOT,        DR750$B_SLOT:
                    00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                              0033                .IIF    NB,DR750$B_BR,  DR750$B_BR:
                    00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                              0034                .IIF    NB,DR750$B_SELF,        DR750$B_SELF:
                    00000035  0034                .IIF    NB,.BLKB,       .BLKB   1
                              0035                .IIF    NB,DR750$B_CONFIG,      DR750$B_CONFIG:
                    00000036  0035                .IIF    NB,.BLKB,       .BLKB   1
                              0036                .IIF    NB,DR750$B_UUT, DR750$B_UUT:
                    00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                              0037                .IIF    NB,DR750$K_LEN, DR750$K_LEN:
                          000007C3                .RESTORE
              30 35 37 52 44 00'  07C3            .ASCIC  "DR750" ; DR750 name
                              05  07C3
                              FF  07C9            .BYTE   255             ; Maximum unit number
                              80  07CA            .BYTE   ^X80            ; Constant to identify start of descriptor
                              82  07CB            .BYTE   ^X82            ; Indicate scan decimal
                 54 4F 4C 53 00'  07CC            .ASCIC  "SLOT"  ; SLOT string
                              04  07CC
              0000000F 0000000A   07D1            .LONG   10,15   ; 10 , 15 limits on input
                              82  07D9            .BYTE   ^X82            ; Indicate scan decimal
                    52 42 00'     07DA            .ASCIC  "BR"    ; BR string
                              02  07DA
              00000007 00000004   07DD            .LONG   4,7     ; 4 , 7 limits on input
                              85  07E5            .BYTE   ^X85            ; For APT, pretend this is a STRING
     54 53 45 54 20 46 4C 45 53 00'  07E6        .ASCIC  "SELF TEST"
                              09  07E6
                    4F 4E 00'     07F0            .ASCIC  "NO"
                              02  07F0
```

```
                        53 45 59 00' 07F3               .ASCIC  "YES"
                                 03  07F3
                                 00  07F7               .BYTE   0
                                 85  07F8               .BYTE   ^X85            ; Indicate scan and verify string
47 49 46 4E 4F 43 20 54 53 45 54 00' 07F9               .ASCIC  "TEST CONFIG"   ; TEST CONFIG string
                                 0B  07F9
                     55 50 43 31 00' 0805               .ASCIC  "1CPU"
                                 04  0805
                     55 50 43 32 00' 080A               .ASCIC  "2CPU"
                                 04  080A
                                 00  080F               .BYTE   0               ; Null length is end of valid 1CPU,2CPU
                                 85  0810               .BYTE   ^X85            ; For APT, pretend this is a STRING
      54 49 4E 55 20 54 53 45 54 00' 0811               .ASCIC  "TEST UNIT"
                                 09  0811
                           4F 4E 00' C81B               .ASCIC  "NO"
                                 02  081B
                        53 45 59 00' 081E               .ASCIC  "YES"
                                 03  081E
                                 00  0822               .BYTE   0
                                 81  0823               .BYTE   ^X81            ; Indicate end of PT-descriptor
                                     0824   255 DR780:   $DS_DR780
                                     0824               .SAVE   LOCAL_BLOCK
                                     0824               .PSECT  $ABS$,ABS
                           00000032  003A               .=HP$A_DEPENDENT
                                     0032               .IIF    NB,DR780$B_TR,  DR780$B_TR:
                           00000033  0032               .IIF    NB,.BLKB,       .BLKB   1
                                     0033               .IIF    NB,DR780$B_BR,  DR780$B_BR:
                           00000034  0033               .IIF    NB,.BLKB,       .BLKB   1
                                     0034               .IIF    NB,DR780$B_SELF,        DR780$B_SELF:
                           00000035  0034               .IIF    NB,.BLKB,       .BLKB   1
                                     0035               .IIF    NB,DR780$B_CONFIG,      DR780$B_CONFIG:
                           00000036  0035               .IIF    NB,.BLKB,       .BLKB   1
                                     0036               .IIF    NB,DR780$B_UUT, DR780$B_UUT:
                           00000037  0036               .IIF    NB,.BLKB,       .BLKB   1
                                     0037               .IIF    NB,DR780$K_LEN, DR780$K_LEN:
                                 00000824               .RESTORE
                  30 38 37 52 44 00' 0824               .ASCIC  "DR780" ; DR780 name
                                 05  0824
                                 FF  082A               .BYTE   255             ; Maximum unit number
                                 80  082B               .BYTE   ^X80            ; Constant to identify start of descriptor
                                 82  082C               .BYTE   ^X82            ; Indicate scan decimal
                           52 54 00' 082D               .ASCIC  "TR"    ; TR string
                                 02  082D
                  0000000F 00000001  0830               .LONG   1,15    ; 1 , 15 limits on input
                                 82  0838               .BYTE   ^X82            ; Indicate scan decimal
                           52 42 00' 0839               .ASCIC  "BR"    ; BR string
                                 02  0839
                  00000007 00000004  083C               .LONG   4,7     ; 4 , 7 limits on input
                                 85  0844               .BYTE   ^X85            ; For APT, pretend this is a STRING
   54 53 45 54 20 46 4C 45 53 00' 0845               .ASCIC  "SELF TEST"
                                 09  0845
                           4F 4E 00' 084F               .ASCIC  "NO"
                                 02  084F
                        53 45 59 00' 0852               .ASCIC  "YES"
                                 03  0852
                                 00  0856               .BYTE   0
                                 85  0857               .BYTE   ^X85            ; Indicate scan and verify string
```

```
47 49 46 4E 4F 43 20 54 53 45 54 00' 0858                .ASCIC  "TEST CONFIG"   ; TEST CONFIG string
                                 0B  0858
                     55 50 43 31 00' 0864                .ASCIC  "1CPU"
                                 04  0864
                     55 50 43 32 00' 0869                .ASCIC  "2CPU"
                                 04  0869
                                 00  086E                .BYTE   0               ; Null length is end of valid 1CPU,2CPU
                                 85  086F                .BYTE   ^X85            ; For APT, pretend this is a STRING
      54 49 4E 55 20 54 53 45 54 00' 0870                .ASCIC  "TEST UNIT"
                                 09  0870
                        4F 4E 00' 087A                  .ASCIC  "NO"
                                 02  087A
                     53 45 59 00' 087D                  .ASCIC  "YES"
                                 03  087D
                                 00  C881                .BYTE   0
                                 81  0882                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                     0883    256 DUP11:  $DS_DUP11
                                     0883                .SAVE   LOCAL_BLOCK
                                     0883                .PSECT  $ABS$,ABS
                            00000032 003A                .=HP$A_DEPENDENT
                                     0032                .IIF    NB,DUP11$L_CSR, DUP11$L_CSR:
                            00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                                     0036                .IIF    NB,DUP11$B_BR,  DUP11$B_BR:
                            00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                                     0037                .IIF    NB,DUP11$K_LEN, DUP11$K_LEN:
                                 00000883                .RESTORE
                  31 31 50 55 44 00' 0883                .ASCIC  "DUP11" ; DUP11 name
                                 05  0883
                                 00  0889                .BYTE   0               ; Maximum unit number
                                 80  088A                .BYTE   ^X80            ; Constant to identify start of descriptor
                                 83  088B                .BYTE   ^X83            ; Indicate scan octal
                        52 53 43 00' 088C                .ASCIC  "CSR"   ; CSR string
                                 03  088C
                     0003FFFE 0003E000  0890             .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                                 83  0898                .BYTE   ^X83            ; Indicate scan octal
                  52 4F 54 43 45 56 00' 0899             .ASCIC  "VECTOR"        ; VECTOR string
                                 06  0899
                     000001FE 00000002  08A0             .LONG   ^02,^0776       ; 2 , 776 limits on input
                                 82  08A8                .BYTE   ^X82            ; Indicate scan decimal
                           52 42 00' 08A9                .ASCIC  "BR"    ; BR string
                                 02  08A9
                     00000007 00000004  08AC             .LONG   4,7     ; 4 , 7 limits on input
                                 81  08B4                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                     08B5    257 DV11:   $DS_DV11
                                     08B5                .SAVE   LOCAL_BLOCK
                                     08B5                .PSECT  $ABS$,ABS
                            00000032 003A                .=HP$A_DEPENDENT
                                     0032                .IIF    NB,DV11$L_CSR,  DV11$L_CSR:
                            00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                                     0036                .IIF    NB,DV11$B_BR,   DV11$B_BR:
                            00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                                     0037                .IIF    NB,DV11$K_LEN,  DV11$K_LEN:
                                 000008B5                .RESTORE
                     31 31 56 44 00' 08B5                .ASCIC  "DV11"  ; DV11 name
                                 04  08B5
                                 00  08BA                .BYTE   0               ; Maximum unit number
                                 80  08BB                .BYTE   ^X80            ; Constant to identify start of descriptor
```

```
                        83  08BC                .BYTE   ^X83              ; Indicate scan octal
                52 53 43 00' 08BD                .ASCIC  "CSR"   ; CSR string
                        03  08BD
           0003FFFE 0003E000  08C1                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        83  08C9                .BYTE   ^X83              ; Indicate scan octal
      52 4F 54 43 45 56 00' 08CA                .ASCIC  "VECTOR"        ; VECTOR string
                        06  08CA
           000001FE 00000002  08D1                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                        82  08D9                .BYTE   ^X82              ; Indicate scan decimal
                52 42 00'   08DA                .ASCIC  "BR"    ; BR string
                        02  08DA
           00000007 00000004  08DD                .LONG   4,7     ; 4 , 7 limits on input
                        81  08E5                .BYTE   ^X81              ; Indicate end of PI-descriptor
                            08E6    258 DW730:  $DS_DW730                                                       ;  [02]
                            08E6                .SAVE   LOCAL_BLOCK
                            08E6                .PSECT  $ABS$,ABS
                    00000032  003A              .=HP$A_DEPENDENT
                              0032              .IIF    NB,DW730$K_LEN, DW730$K_LEN:
                            08E6                .RESTORE
      30 33 37 57 44 00' 08E6                   .ASCIC  "DW730" ; DW730 name
                        05  08E6
                        01  08EC                .BYTE   1                 ; Maximum unit number
                        80  08ED                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  08EE                .BYTE   ^X81              ; Indicate end of PI-descriptor
                            08EF    259 DW750:  $DS_DW750
                            08EF                .SAVE   LOCAL_BLOCK
                            08EF                .PSECT  $ABS$,ABS
                    00000032  003A              .=HP$A_DEPENDENT
                              0032              .IIF    NB,DW750$K_LEN, DW750$K_LEN:
                            08EF                .RESTORE
   30 35 37 57 44 00' 08EF                      .ASCIC  "DW750" ; DW750 name
                        05  08EF
                        04  08F5                .BYTE   4                 ; Maximum unit number
                        80  08F6                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  08F7                .BYTE   ^X81              ; Indicate end of PI-descriptor
                            08F8    260 DW780:  $DS_DW780
                            08F8                .SAVE   LOCAL_BLOCK
                            08F8                .PSECT  $ABS$,ABS
                    00000032  003A              .=HP$A_DEPENDENT
                              0032              .IIF    NB,DW780$B_TR,  DW780$B_TR:
                    00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                              0033              .IIF    NB,DW780$B_BR,  DW780$B_BR:
                    00000034  0033              .IIF    NB,.BLKB,       .BLKB   1
                              0034              .IIF    NB,DW780$K_LEN, DW780$K_LEN:
                            08F8                .RESTORE
      30 38 37 57 44 00' 08F8                   .ASCIC  "DW780" ; DW780 name
                        05  08F8
                        08  08FE                .BYTE   8                 ; Maximum unit number
                        80  08FF                .BYTE   ^X80              ; Constant to identify start of descriptor
                        82  0900                .BYTE   ^X82              ; Indicate scan decimal
                52 54 00'   0901                .ASCIC  "TR"    ; TR string
                        02  0901
           0000000F 00000001  0904                .LONG   1,15    ; 1 , 15 limits on input
                        82  090C                .BYTE   ^X82              ; Indicate scan decimal
                52 42 00'   090D                .ASCIC  "BR"    ; BR string
                        02  090D
           00000007 00000004  0910                .LONG   4,7     ; 4 , 7 limits on input
```

```
                      81  0918                    .BYTE   ^X81            ; Indicate end of PT-descriptor
                          0919    261 DZ11:       $DS_DZ11
                          0919                    .SAVE   LOCAL_BLOCK
                          0919                    .PSECT  $ABS$,ABS
                 00000032 003A                    .=HP$A_DEPENDENT
                          0032                    .IIF    NB,DZ11$L_CSR,  DZ11$L_CSR:
                 00000036 0032                    .IIF    NB,.BLKL,       .BLKL   1
                          0036                    .IIF    NB,DZ11$B_BR,   DZ11$B_BR:
                 00000037 0036                    .IIF    NB,.BLKB,       .BLKB   1
                          0037                    .IIF    NB,DZ11$B_MTYPE,        DZ11$B_MTYPE:
                 00000038 0037                    .IIF    NB,.BLKB,       .BLKB   1
                          0038                    .IIF    NB,DZ11$K_LEN,  DZ11$K_LEN:
                 00000919                         .RESTORE
           31 31 5A 44 00' 0919                   .ASCIC  "DZ11"  ; DZ11 name
                      04  0919
                      00  091E                    .BYTE   0               ; Maximum unit number
                      80  091F                    .BYTE   ^X80            ; Constant to identify start of descriptor
                      83  0920                    .BYTE   ^X83            ; Indicate scan octal
              52 53 43 00' 0921                   .ASCIC  "CSR"   ; CSR string
                      03  0921
          0003FFFE 0003E000  0925                 .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      83  092D                    .BYTE   ^X83            ; Indicate scan octal
        52 4F 54 43 45 56 00' 092E                .ASCIC  "VECTOR"        ; VECTOR string
                      06  092E
          000001FE 00000002  0935                 .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      82  093D                    .BYTE   ^X82            ; Indicate scan decimal
                 52 42 00' 093E                   .ASCIC  "BR"    ; BR string
                      02  093E
          00000007 00000004  0941                 .LONG   4,7     ; 4 , 7 limits on input
                      85  0949                    .BYTE   ^X85            ; Indicate scan and verify string
45 50 59 54 20 45 4C 55 44 4F 4D 00' 094A         .ASCIC  "MODULE TYPE"   ; MODULE TYPE string
                      0B  094A
                 41 49 45 00' 0956                .ASCIC  "EIA"
                      03  0956
              41 4D 30 32 00' 095A                .ASCIC  "20MA"
                      04  095A
                      00  095F                    .BYTE   0               ; Null length is end of valid EIA,20MA
                      81  0960                    .BYTE   ^X81            ; Indicate end of PT-descriptor
                          0961    262 DZ32:       $DS_DZ32                                                        ;       [08]
                          0961                    .SAVE   LOCAL_BLOCK
                          0961                    .PSECT  $ABS$,ABS
                 00000032 003A                    .=HP$A_DEPENDENT
                          0032                    .IIF    NB,HP$B_DZ32_BRLVL,     HP$B_DZ32_BRLVL:
                          0032                    .IIF    NB,DZ32$B_BRLVL,        DZ32$B_BRLVL:
                 00000033 0032                    .IIF    NB,.BLKB,       .BLKB   1
                          0033                    .IIF    NB,HP$B_DZ32_ACTIV,     HP$B_DZ32_ACTIV:
                          0033                    .IIF    NB,DZ32$B_ACTIV,        DZ32$B_ACTIV:
                 00000034 0033                    .IIF    NB,.BLKB,       .BLKB   1
                          0034                    .IIF    NB,HP$B_DZ32_SPEED,     HP$B_DZ32_SPEED:
                          0034                    .IIF    NB,DZ32$B_SPEED,        DZ32$B_SPEED:
                 00000035 0034                    .IIF    NB,.BLKB,       .BLKB   1
                          0035                    .IIF    NB,HP$W_DZ32_FLAGS,     HP$W_DZ32_FLAGS:
                          0035                    .IIF    NB,DZ32$W_FLAGS,        DZ32$W_FLAGS:
                 00000037 0035                    .IIF    NB,.BLKW,       .BLKW   1
                          0037                    .IIF    NB,HP$K_LEN,    HP$K_LEN:
                          0037                    .IIF    NB,DZ32$_LEN,   DZ32$_LEN:
                 00000961                         .RESTORE
```

```
                     32 33 5A 44 00' 0961              .ASCIC  "DZ32"  ; DZ32 name
                                 04  0961
                                 00  0966              .BYTE   0               ; Maximum unit number
                                 80  0967              .BYTE   ^X80            ; Constant to identify start of descriptor
                                 83  0968              .BYTE   ^X83            ; Indicate scan octal
                        52 53 43 00' 0969              .ASCIC  "CSR"   ; CSR string
                                 03  0969
                  0003FFFE 0003E000  096D              .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                 83  0975              .BYTE   ^X83            ; Indicate scan octal
               52 4F 54 43 45 56 00' 0976              .ASCIC  "VECTOR"        ; VECTOR string
                                 06  0976
                  000001FE 00000002  097D              .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                 83  0985              .BYTE   ^X83            ; Indicate scan octal
                           52 42 00' 0986              .ASCIC  "BR"    ; BR string
                                 02  0986
                  00000007 00000004  0989              .LONG   ^O4,^O7 ; 4 , 7 limits on input
                                 83  0991              .BYTE   ^X83            ; Indicate scan octal
45 4E 49 4C 20 45 56 49 54 43 41 00' 0992              .ASCIC  "ACTIVE LINES"  ; ACTIVE LINES string
                                 53  099E
                                 0C  0992
                  000000FF 00000000  099F              .LONG   ^O0,^O377       ; 0 , 377 limits on input
                                 83  09A7              .BYTE   ^X83            ; Indicate scan octal
      45 54 41 52 20 44 55 41 42 00' 09A8              .ASCIC  "BAUD RATE"     ; BAUD RATE string
                                 09  09A8
                  0000000E 00000000  09B2              .LONG   ^O0,^O16        ; 0 , 16 limits on input
                                 83  09BA              .BYTE   ^X83            ; Indicate scan octal
59 54 20 4B 43 41 42 50 4F 4F 4C 00' 09BB              .ASCIC  "LOOPBACK TYPE" ; LOOPBACK TYPE string
                              45 50  09C7
                                 0D  09BB
                  00000004 00000000  09C9              .LONG   ^O0,^O4 ; 0 , 4 limits on input
                                 83  09D1              .BYTE   ^X83            ; Indicate scan octal
53 20 54 45 53 45 52 20 48 4E 49 00' 09D2              .ASCIC  "INH RESET SWITCH"      ; INH RESET SWITCH string
                     48 43 54 49 57  09DE
                                 10  09D2
                  00000001 00000000  09E3              .LONG   ^O0,^O1 ; 0 , 1 limits on input
                                 83  09EB              .BYTE   ^X83            ; Indicate scan octal
48 43 54 49 57 53 20 45 44 4F 4D 00' 09EC              .ASCIC  "MODE SWITCH"   ; MODE SWITCH string
                                 0B  09EC
                  00000001 00000000  09F8              .LONG   ^O0,^O1 ; 0 , 1 limits on input
                                 83  0A00              .BYTE   ^X83            ; Indicate scan octal
44 45 45 50 53 20 54 49 4C 50 53 00' 0A01              .ASCIC  "SPLIT SPEED JUMPER"    ; SPLIT SPEED JUMPER string
                  52 45 50 4D 55 4A 20  0A0D
                                 12  0A01
                  00000002 00000000  0A14              .LONG   ^O0,^O2 ; 0 , 2 limits on input
                                 81  0A1C              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                     0A1D  263 DZV11:  $DS_DZV11                                       ;  [19]
                                     0A1D              .SAVE   LOCAL_BLOCK
                                     0A1D              .PSECT  $ABS$,ABS
                           00000032  003A              .=HP$A_DEPENDENT
                                     0032              .IIF    NB,DZV11$L_CSR, DZV11$L_CSR:
                           00000036  0032              .IIF    NB,,BLKL,       .BLKL   1
                                     0036              .IIF    NB,DZV11$K_LEN, DZV11$K_LEN:
                                 00000A1D              .RESTORE
                  31 31 56 5A 44 00' 0A1D              .ASCIC  "DZV11" ; DZV11 name
                                 05  0A1D
                                 00  0A23              .BYTE   0               ; Maximum unit number
                                 80  0A24              .BYTE   ^X80            ; Constant to identify start of descriptor
```

```
                        83  0A25                .BYTE   ^X83              ; Indicate scan octal
                52 53 43 00' 0A26                .ASCIC  "CSR"   ; CSR string
                        03  0A26
           0003FFFE 0003E000  0A2A                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        83  0A32                .BYTE   ^X83              ; Indicate scan octal
        52 4F 54 43 45 56 00' 0A33                .ASCIC  "VECTOR"          ; VECTOR string
                        06  0A33
           000001FE 00000002  0A3A                .LONG   ^O2,^O776         ; 2 , 776 limits on input
                        81  0A42                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            0A43    264 IEU11A: $DS_IEU11A                              ;                                       [16]
                            0A43                .SAVE   LOCAL_BLOCK
                            0A43                .PSECT  $ABS$,ABS
                    00000032  003A                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$B_IEU11A_BR,      HP$B_IEU11A_BR:
                            C032                .IIF    NB,IEU11A$B_BR, IEU11A$B_BR:
                    00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                            0033                .IIF    NB,HP$K_IEU11A_LEN,     HP$K_IEU11A_LEN:
                            0033                .IIF    NB,IEU11A$K_LEN,        IEU11A$K_LEN:
                    00000A43                    .RESTORE
        41 31 31 55 45 49 00' 0A43                .ASCIC  "IEU11A"          ; IEU11A name
                        06  0A43
                        01  0A4A                .BYTE   1                 ; Maximum unit number
                        80  0A4B                .BYTE   ^X80              ; Constant to identify start of descriptor
                        83  0A4C                .BYTE   ^X83              ; Indicate scan octal
              65 73 61 42 00' 0A4D                .ASCIC  "Base"  ; Base string
                        04  0A4D
           0003FFF0 0003E000  0A52                .LONG   ^O760000,^O777760       ; 760000 , 777760 limits on input
                        83  0A5A                .BYTE   ^X83              ; Indicate scan octal
        72 6F 74 63 65 56 00' 0A5B                .ASCIC  "Vector"          ; Vector string
                        06  0A5B
           000001FC 00000000  0A62                .LONG   ^O0,^O774         ; 0 , 774 limits on input
                        82  0A6A                .BYTE   ^X82              ; Indicate scan decimal
                52 42 00' 0A6B                .ASCIC  "BR"    ; BR string
                        02  0A6B
           00000007 00000004  0A6E                .LONG   4,7       ; 4 , 7 limits on input
                        81  0A76                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            0A77    265         $DS_KA_DEF
                            0A77                .SAVE   LOCAL_BLOCK
                            0A77                .PSECT  $ABS$,ABS
                    00000032  003A                .=HP$A_DEPENDENT
                            0032                .IIF    NB,KA$L_FLAGS,  KA$L_FLAGS:
                    00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                            0036                .IIF    NB,KA$W_WCS,    KA$W_WCS:
                    00000038  0036                .IIF    NB,.BLKW,       .BLKW   1
                            0038                .IIF    NB,KA$W_RESERVED,       KA$W_RESERVED:
                    0000003A  0038                .IIF    NB,.BLKW,       .BLKW   1
                            003A                .IIF    NB,KA$L_SID,    KA$L_SID:
                    0000003D  003A                .IIF    NB,.BLKB,       .BLKB   3
                            003D                .IIF    NB,KA$B_SID_TYPE,       KA$B_SID_TYPE:
                    0000003E  003D                .IIF    NB,.BLKB,       .BLKB   1
                            003E                .IIF    NB,KA$W_LATENCY,        KA$W_LATENCY:
                    00000040  003E                .IIF    NB,.BLKW,       .BLKW   1
                            0040                .IIF    NB,KA$W_PROGRESS,       KA$W_PROGRESS:
                    00000042  0040                .IIF    NB,.BLKW,       .BLKW   1
                            0042                .IIF    NB,KA$B_ACC_TYPE,       KA$B_ACC_TYPE:
                    00000043  0042                .IIF    NB,.BLKB,       .BLKB   1
                            0043                .IIF    NB,KA$B_RESERVED,       KA$B_RESERVED:
```

```
                                00000044  0043                .IIF    NB,.BLKB,          .BLKB   1
                                          0044                .IIF    NB,KA$W_MEM_SIZE,          KA$W_MEM_SIZE:
                                00000046  0044                .IIF    NB,.BLKW,          .BLKW   1
                                          0046                .IIF    NB,KA$K_LEN,       KA$K_LEN:
                                   00000A77                   .RESTORE
                                          0A77    266 KA730:  $DS_KA730                  ;                                       [01]
                        30 33 37 41 4B 00' 0A77               .ASCIC  "KA730" ; KA730 name
                                       05  0A77
                                       04  0A7D               .BYTE   4                  ; Maximum unit number
                                       80  0A7E               .BYTE   ^X80               ; Constant to identify start of descriptor
                                       85  0A7F               .BYTE   ^X85               ; For APT, pretend this is a STRING
61 65 79 2D 66 6F 2D 65 6D 69 54 00' 0A80                     .ASCIC  "Time-of-year clock"
                  6B 63 6F 6C 63 20  72  0A8C
                                       12  0A80
                           4F 4E 00' CA93                     .ASCIC  "NO"
                                       02  0A93
                        53 45 59 00' 0A96                     .ASCIC  "YES"
                                       03  0A96
                                       00  0A9A               .BYTE   0
                                       84  0A9B               .BYTE   ^X84               ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00' 0A9C                     .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64  0AA8
                                       10  0A9C
                  0000FFFF 00000000  0AAD                     .LONG   0,1@16-1           ; 0 , 1@16-1 limits on input
                                       82  0AB5               .BYTE   ^X82               ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 0AB6                     .ASCIC  "Accelerator type"      ; Accelerator type string
                     65 70 79 74 20  0AC2
                                       10  0AB6
                  000000FF 0C000000  0AC7                     .LONG   0,255   ; 0 , 255 limits on input
                                       82  0ACF               .BYTE   ^X82               ; Indicate scan decimal
20 66 6F 20 73 65 74 79 62 2D 4B 00' 0AD0                     .ASCIC  "K-bytes of Main Memory"        ; K-bytes of Main Memory string
   79 72 6F 6D 65 4D 20 6E 69 61 4D  0ADC
                                       16  0AD0
                  00001400 00000000  0AE7                     .LONG   0,5120  ; 0 , 5120 limits on input
                                       85  0AEF               .BYTE   ^X85               ; For APT, pretend this is a STRING
6F 6C 20 53 43 57 20 72 65 73 55 00' 0AF0                     .ASCIC  "User WCS loaded"
                           64 65 64 61  0AFC
                                       0F  0AF0
                              4F 4E 00' 0B00                  .ASCIC  "NO"
                                       02  0B00
                           53 45 59 00' 0B03                  .ASCIC  "YES"
                                       03  0B03
                                       00  0B07               .BYTE   0
                                       85  0B08               .BYTE   ^X85               ; For APT, pretend this is a STRING
      73 72 6F 72 72 65 20 42 53 00' 0B09                     .ASCIC  "SB errors"
                                       09  0B09
                              4F 4E 00' 0B13                  .ASCIC  "NO"
                                       02  0B13
                           53 45 59 00' 0B16                  .ASCIC  "YES"
                                       03  0B16
                                       00  0B1A               .BYTE   0
                                       81  0B1B               .BYTE   ^X81               ; Indicate end of PT-descriptor
                                          0B1C    267 KA750:  $DS_KA750
                        30 35 37 41 4B 00' 0B1C               .ASCIC  "KA750" ; KA750 name
                                       05  0B1C
                                       04  0B22               .BYTE   4                  ; Maximum unit number
                                       80  0B23               .BYTE   ^X80               ; Constant to identify start of descriptor
```

```
                                                85  0B24                .BYTE   ^X85              ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 47 00'            0B25                .ASCIC  "G-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69             0B31
                                                17  0B25
                                     4F 4E 00'  0B3D                .ASCIC  "NO"
                                                02  0B3D
                                  53 45 59 00'  0B40                .ASCIC  "YES"
                                                03  0B40
                                                00  0B44                .BYTE   0
                                                85  0B45                .BYTE   ^X85              ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 48 00'            0B46                .ASCIC  "H-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69             0B52
                                                17  0B46
                                     4F 4E 00'  0B5E                .ASCIC  "NO"
                                                02  0B5E
                                  53 45 59 00'  0B61                .ASCIC  "YES"
                                                03  0B61
                                                00  0B65                .BYTE   0
                                                85  0B66                .BYTE   ^X85              ; For APT, pretend this is a STRING
61 65 79 2D 66 6F 2D 65 6D 69 54 00'            0B67                .ASCIC  "Time-of-year clock"
                  6B 63 6F 6C 63 20 72          0B73
                                                12  0B67
                                     4F 4E 00'  0B7A                .ASCIC  "NO"
                                                02  0B7A
                                  53 45 59 00'  0B7D                .ASCIC  "YES"
                                                03  0B7D
                                                00  0B81                .BYTE   0
                                                84  0B82                .BYTE   ^X84              ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'            0B83                .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64             0B8F
                                                10  0B83
                  0000FFFF 00000000             0B94                .LONG   0,1@16-1          ; 0 , 1@16-1 limits on input
                                                82  0B9C                .BYTE   ^X82              ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'            0B9D                .ASCIC  "Accelerator type"      ; Accelerator type string
                     65 70 79 74 20             0BA9
                                                10  0B9D
                  000000FF 00000000             0BAE                .LONG   0,255   ; 0 , 255 limits on input
                                                81  0BB6                .BYTE   ^X81              ; Indicate end of PT-descriptor
                                                    0BB7    268 KA780:  $DS_KA780
                  30 38 37 41 4B 00'            0BB7                .ASCIC  "KA780" ; KA780 name
                                                05  0BB7
                                                04  0BBD                .BYTE   4                 ; Maximum unit number
                                                80  0BBE                .BYTE   ^X80              ; Constant to identify start of descriptor
                                                85  0BBF                .BYTE   ^X85              ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 47 00'            0BC0                .ASCIC  "G-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69             0BCC
                                                17  0BC0
                                     4F 4E 00'  0BD8                .ASCIC  "NO"
                                                02  0BD8
                                  53 45 59 00'  0BDB                .ASCIC  "YES"
                                                03  0BDB
                                                00  0BDF                .BYTE   0
                                                85  0BE0                .BYTE   ^X85              ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 48 00'            0BE1                .ASCIC  "H-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69             0BED
                                                17  0BE1
                                     4F 4E 00'  0BF9                .ASCIC  "NO"
```

```
                                  02  0BF9
                           53 45 59 00' 0BFC                .ASCIC  "YES"
                                  03  0BFC
                                  00  0C00                .BYTE   0
                                  84  0C01                .BYTE   ^X84            ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00' 0C02                .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64  0C0E
                                  10  0C02
                        0000FFFF 00000000  0C13          .LONG   0,1@16-1        ; 0 , 1@16-1 limits on input
                                  82  0C1B                .BYTE   ^X82            ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 0C1C                .ASCIC  "Accelerator type"      ; Accelerator type string
                     65 70 79 74 20  0C28
                                  10  0C1C
                        000000FF 00000000  0C2D          .LONG   0,255   ; 0 , 255 limits on input
                                  81  0C35                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                      0C36  269 KAXXX:    $DS_KAXXX                       ;                       [15]
                           58 58 58 41 4B 00' 0C36       .ASCIC  "KAXXX" ; KAXXX name
                                  05  0C36
                                  04  0C3C                .BYTE   4               ; Maximum unit number
                                  80  0C3D                .BYTE   ^X80            ; Constant to identify start of descriptor
                                  85  0C3E                .BYTE   ^X85            ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 47 00' 0C3F                .ASCIC  "G-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69  0C4B
                                  17  0C3F
                              4F 4E 00' 0C57             .ASCIC  "NO"
                                  02  0C57
                           53 45 59 00' 0C5A             .ASCIC  "YES"
                                  03  0C5A
                                  00  0C5E                .BYTE   0
                                  85  0C5F                .BYTE   ^X85            ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 48 00' 0C60                .ASCIC  "H-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69  0C6C
                                  17  0C60
                              4F 4E 00' 0C78             .ASCIC  "NO"
                                  02  0C78
                           53 45 59 00' 0C7B             .ASCIC  "YES"
                                  03  0C7B
                                  00  0C7F                .BYTE   0
                                  84  0C80                .BYTE   ^X84            ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00' 0C81                .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64  0C8D
                                  10  0C81
                        0000FFFF 00000000  0C92          .LONG   0,1@16-1        ; 0 , 1@16-1 limits on input
                                  82  0C9A                .BYTE   ^X82            ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 0C9B                .ASCIC  "Accelerator type"      ; Accelerator type string
                     65 70 79 74 20  0CA7
                                  10  0C9B
                        000000FF 00000000  0CAC          .LONG   0,255   ; 0 , 255 limits on input
                                  81  0CB4                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                      0CB5  270 KA785:    $DS_KA785                       ;                       [15]
                           35 38 37 41 4B 00' 0CB5       .ASCIC  "KA785" ; KA785 name
                                  05  0CB5
                                  04  0CBB                .BYTE   4               ; Maximum unit number
                                  80  0CBC                .BYTE   ^X80            ; Constant to identify start of descriptor
                                  85  0CBD                .BYTE   ^X85            ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 47 00' 0CBE                .ASCIC  "G-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69  0CCA
```

```
                                    17  0CBE
                                4F 4E 00' 0CD6              .ASCIC  "NO"
                                    02  0CD6
                             53 45 59 00' 0CD9              .ASCIC  "YES"
                                    03  0CD9
                                    00  0CDD              .BYTE   0
                                    85  0CDE              .BYTE   ^X85            ; For APT, pretend this is a STRING
20 67 6E 69 74 61 6F 6C 66 2D 48 00' 0CDF              .ASCIC  "H-floating instructions"
73 6E 6F 69 74 63 75 72 74 73 6E 69  0CEB
                                    17  0CDF
                                4F 4E 00' 0CF7              .ASCIC  "NO"
                                    02  0CF7
                             53 45 59 00' 0CFA              .ASCIC  "YES"
                                    03  0CFA
                                    00  0CFE              .BYTE   0
                                    84  0CFF              .BYTE   ^X84            ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 4A 00' 0D00              .ASCIC  "JCS last address"      ; JCS last address string
                             73 73 65 72 64  0D0C
                                    10  0D00
                         0000FFFF 00000000  0D11              .LONG   0,1@16-1        ; 0 , 1@16-1 limits on input
                                    82  0D19              .BYTE   ^X82            ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 0D1A              .ASCIC  "Accelerator type"      ; Accelerator type string
                             65 70 79 74 20  0D26
                                    10  0D1A
                         000000FF 00000000  0D2B              .LONG   0,255   ; 0 , 255 limits on input
                                    81  0D33              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                        0D34    271 KMC11:  $DS_KMC11
                                        0D34                .SAVE   LOCAL_BLOCK
                                        0D34                .PSECT  $ABS$,ABS
                                 00000032  0046             .=HP$A_DEPENDENT
                                        0032                .IIF    NB,KMC11$L_CSR, KMC11$L_CSR:
                                 00000036  0032             .IIF    NB,.BLKL,       .BLKL   1
                                        0036                .IIF    NB,KMC11$B_BR,  KMC11$B_BR:
                                 00000037  0036             .IIF    NB,.BLKB,       .BLKB   1
                                        0037                .IIF    NB,KMC11$K_LEN, KMC11$K_LEN:
                                   00000D34                 .RESTORE
                           31 31 43 4D 4B 00' 0D34          .ASCIC  "KMC11" ; KMC11 name
                                    05  0D34
                                    00  0D3A              .BYTE   0               ; Maximum unit number
                                    80  0D3B              .BYTE   ^X80            ; Constant to identify start of descriptor
                                    83  0D3C              .BYTE   ^X83            ; Indicate scan octal
                                 52 53 43 00' 0D3D          .ASCIC  "CSR"   ; CSR string
                                    03  0D3D
                           0003FFFE 0003E000  0D41          .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                                    83  0D49              .BYTE   ^X83            ; Indicate scan octal
                        52 4F 54 43 45 56 00' 0D4A          .ASCIC  "VECTOR"        ; VECTOR string
                                    06  0D4A
                           000001FE 00000002  0D51          .LONG   ^02,^0776       ; 2 , 776 limits on input
                                    82  0D59              .BYTE   ^X82            ; Indicate scan decimal
                                 52 42 00' 0D5A             .ASCIC  "BR"    ; BR string
                                    02  0D5A
                           00000007 00000004  0D5D          .LONG   4,7     ; 4 , 7 limits on input
                                    81  0D65              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                        0D66    272 KW11K:  $DS_KW11K
                                        0D66                .SAVE   LOCAL_BLOCK
                                        0D66                .PSECT  $ABS$,ABS
                                 00000032  0046             .=HP$A_DEPENDENT
```

```
                                    0032                .IIF    NB,KW11K$B_BR,  KW11K$B_BR:
                           00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                                    0033                .IIF    NB,KW11K$K_LEN, KW11K$K_LEN:
                                00000D66                .RESTORE
                     4B 31 31 57 4B 00' 0D66            .ASCIC  "KW11K" ; KW11K name
                                    05  0D66
                                    02  0D6C            .BYTE   2               ; Maximum unit number
                                    80  0D6D            .BYTE   ^X80            ; Constant to identify start of descriptor
                                    83  0D6E            .BYTE   ^X83            ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 0D6F            .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                                    0B  0D6F
                        0003FFFE 0003E000  0D7B      .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                    83  0D83            .BYTE   ^X83            ; Indicate scan octal
                     52 4F 54 43 45 56 00' 0D84      .ASCIC  "VECTOR"        ; VECTOR string
                                    06  0D84
                        000001FE 00000002  0D8B      .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                    82  0D93            .BYTE   ^X82            ; Indicate scan decimal
                              52 42 00' 0D94            .ASCIC  "BR"    ; BR string
                                    02  0D94
                        00000007 00000004  0D97      .LONG   4,7     ; 4 , 7 limits on input
                                    81  0D9F            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0DA0    273 LA12:   $DS_LA12                        ;                       [14]
                                    0DA0                .SAVE   LOCAL_BLOCK
                                    0DA0                .PSECT  $ABS$,ABS
                                00000032  0046          .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$K_LA12_LEN,       HP$K_LA12_LEN:
                                    0032                .IIF    NB,LA12$K_LEN,  LA12$K_LEN:
                                00000DA0                .RESTORE
                        32 31 41 4C 00' 0DA0            .ASCIC  "LA12"  ; LA12 name
                                    04  0DA0
                                    08  0DA5            .BYTE   8               ; Maximum unit number
                                    80  0DA6            .BYTE   ^X80            ; Constant to identify start of descriptor
                                    81  0DA7            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0DA8    274 LA100:  $DS_LA100                       ;                       [14]
                                    0DA8                .SAVE   LOCAL_BLOCK
                                    0DA8                .PSECT  $ABS$,ABS
                                00000032  0046          .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$K_LA100_LEN,      HP$K_LA100_LEN:
                                    0032                .IIF    NB,LA100$K_LEN, LA100$K_LEN:
                                00000DA8                .RESTORE
                     30 30 31 41 4C 00' 0DA8            .ASCIC  "LA100" ; LA100 name
                                    05  0DA8
                                    08  0DAE            .BYTE   8               ; Maximum unit number
                                    80  0DAF            .BYTE   ^X80            ; Constant to identify start of descriptor
                                    81  0DB0            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0DB1    275 LA120:  $DS_LA120
                                    0DB1                .SAVE   LOCAL_BLOCK
                                    0DB1                .PSECT  $ABS$,ABS
                                00000032  0046          .=HP$A_DEPENDENT
                                    0032                .IIF    NB,LA120$K_LEN, LA120$K_LEN:
                                00000DB1                .RESTORE
                     30 32 31 41 4C 00' 0DB1            .ASCIC  "LA120" ; LA120 name
                                    05  0DB1
                                    08  0DB7            .BYTE   8               ; Maximum unit number
                                    80  0DB8            .BYTE   ^X80            ; Constant to identify start of descriptor
                                    81  0DB9            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0DBA    276 LA180:  $DS_LA180
```

```
                        0DBA                    .SAVE   LOCAL_BLOCK
                        0DBA                    .PSECT  $ABS$,ABS
                00000032  0046                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,LA180$K_LEN, LA180$K_LEN:
                    00000DBA                    .RESTORE
      30 38 31 41 4C 00' 0DBA                   .ASCIC  ''LA180'' ; LA180 name
                          05  0DBA
                          01  0DC0             .BYTE   1               ; Maximum unit number
                          80  0DC1             .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  0DC2             .BYTE   ^X81            ; Indicate end of PT-descriptor
                              0DC3  277 LA34:   $DS_LA34
                              0DC3              .SAVE   LOCAL_BLOCK
                              0DC3              .PSECT  $ABS$,ABS
                00000032      0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,LA34$K_LEN,  LA34$K_LEN:
                        00000DC3                .RESTORE
         34 33 41 4C 00' 0DC3                   .ASCIC  ''LA34''  ; LA34 name
                          04  0DC3
                          08  0DC8             .BYTE   8               ; Maximum unit number
                          80  0DC9             .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  0DCA             .BYTE   ^X81            ; Indicate end of PT-descriptor
                              0DCB  278 LA36:   $DS_LA36
                              0DCB              .SAVE   LOCAL_BLOCK
                              0DCB              .PSECT  $ABS$,ABS
                00000032      0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,LA36$K_LEN,  LA36$K_LEN:
                        00000DCB                .RESTORE
         36 33 41 4C 00' 0DCB                   .ASCIC  ''LA36''  ; LA36 name
                          04  0DCB
                          08  0DD0             .BYTE   8               ; Maximum unit number
                          80  0DD1             .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  0DD2             .BYTE   ^X81            ; Indicate end of PT-descriptor
                              0DD3  279 LA38:   $DS_LA38
                              0DD3              .SAVE   LOCAL_BLOCK
                              0DD3              .PSECT  $ABS$,ABS
                00000032      0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,LA38$K_LEN,  LA38$K_LEN:
                        00000DD3                .RESTORE
      38 33 41 4C 00' 0DD3                      .ASCIC  ''LA38''  ; LA38 name
                          04  0DD3
                          08  0DD8             .BYTE   8               ; Maximum unit number
                          80  0DD9             .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  0DDA             .BYTE   ^X81            ; Indicate end of PT-descriptor
                              0DDB  280 LESI:   $DS_LESI                          ;                                   [13]
                              0DDB              .SAVE   LOCAL_BLOCK
                              0DDB              .PSECT  $ABS$,ABS
                00000032      0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,HP$L_LESI_IP,        HP$L_LESI_IP:
                              0032              .IIF    NB,LESI$L_IP,   LESI$L_IP:
                00000036      0032              .IIF    NB,.BLKL,       .BLKL   1
                              0036              .IIF    NB,HP$B_LESI_BR,        HP$B_LESI_BR:
                              0036              .IIF    NB,LESI$B_BR,   LESI$B_BR:
                00000037      0036              .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,HP$K_LESI_LEN,       HP$K_LESI_LEN:
                              0037              .IIF    NB,LESI$K_LEN,  LESI$K_LEN:
                        00000DDB                .RESTORE
      49 53 45 4C 00' 0DDB                      .ASCIC  ''LESI''  ; LESI name
```

```
                                04  0DDB
                                00  0DE0                .BYTE   0                       ; Maximum unit number
                                80  0DE1                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                83  0DE2                .BYTE   ^X83                    ; Indicate scan octal
                          50 49 00' 0DE3                .ASCIC  "IP"    ; IP string
                                02  0DE3
                    0003FFFC 0003E000  0DE6             .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                                83  0DEE                .BYTE   ^X83                    ; Indicate scan octal
                 72 6F 74 63 65 56 00'  0DEF            .ASCIC  "Vector"                ; Vector string
                                06  0DEF
                    000001FC 00000004  0DF6             .LONG   ^O4,^O774               ; 4 , 774 limits on input
                                82  0DFE                .BYTE   ^X82                    ; Indicate scan decimal
                          52 42 00' 0DFF                .ASCIC  "BR"    ; BR string
                                02  0DFF
                    00000007 00000004  CE02             .LONG   4,7     ; 4 , 7 limits on input
                                81  0E0A                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                    0E0B    281 LN01:   $DS_LN01                        ;                                       [14]
                                    0E0B                .SAVE   LOCAL_BLOCK
                                    0E0B                .PSECT  $ABS$,ABS
                          00000032  0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,LN01$K_LEN,  LN01$K_LEN:
                                00000E0B                .RESTORE
                       31 30 4E 4C 00'  0E0B            .ASCIC  "LN01"  ; LN01 name
                                04  0E0B
                                01  0E10                .BYTE   1                       ; Maximum unit number
                                80  0E11                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                81  0E12                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                    0E13    282 LP04:   $DS_LP04
                                    0E13                .SAVE   LOCAL_BLOCK
                                    0E13                .PSECT  $ABS$,ABS
                          00000032  0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,LP04$K_LEN,  LP04$K_LEN:
                                00000E13                .RESTORE
                    34 30 50 4C 00'  0E13               .ASCIC  "LP04"  ; LP04 name
                                04  0E13
                                01  0E18                .BYTE   1                       ; Maximum unit number
                                80  0E19                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                81  0E1A                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                    0E1B    283 LP05:   $DS_LP05
                                    0E1B                .SAVE   LOCAL_BLOCK
                                    0E1B                .PSECT  $ABS$,ABS
                          00000032  0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,LP05$K_LEN,  LP05$K_LEN:
                                00000E1B                .RESTORE
                    35 30 50 4C 00'  0E1B               .ASCIC  "LP05"  ; LP05 name
                                04  0E1B
                                01  0E20                .BYTE   1                       ; Maximum unit number
                                80  0E21                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                81  0E22                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                    0E23    284 LP06:   $DS_LP06
                                    0E23                .SAVE   LOCAL_BLOCK
                                    0E23                .PSECT  $ABS$,ABS
                          00000032  0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,LP06$K_LEN,  LP06$K_LEN:
                                00000E23                .RESTORE
                    36 30 50 4C 00'  0E23               .ASCIC  "LP06"  ; LP06 name
                                04  0E23
```

```
                          01  0E28                .BYTE   1                   ; Maximum unit number
                          80  0E29                .BYTE   ^X80                ; Constant to identify start of descriptor
                          81  0E2A                .BYTE   ^X81                ; Indicate end of PT-descriptor
                              0E2B    285 LP07:   $DS_LP07                        ;                                       [15]
                              0E2B                .SAVE   LOCAL_BLOCK
                              0E2B                .PSECT  $ABS$,ABS
                      00000032  0046              .=HP$A_DEPENDENT
                                0032              .IIF    NB,LP07$K_LEN,  LP07$K_LEN:
                            00000E2B              .RESTORE
                37 30 50 4C 00' 0E2B              .ASCIC  "LP07"  ; LP07 name
                          04  0E2B
                          01  0E30                .BYTE   1                   ; Maximum unit number
                          80  0E31                .BYTE   ^X80                ; Constant to identify start of descriptor
                          81  0E32                .BYTE   ^X81                ; Indicate end of PT-descriptor
                              0E33    286 LP11:   $DS_LP11
                              0E33                .SAVE   LOCAL_BLOCK
                              0E33                .PSECT  $ABS$,ABS
                      000000?2  0046              .=HP$A_DEPENDENT
                                0032              .IIF    NB,LP11$L_CSR,  LP11$L_CSR:
                      00000036  0032              .IIF    NB,.BLKL,       .BLKL   1
                                0036              .IIF    NB,LP11$B_BR,   LP11$B_BR:
                      00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                                0037              .IIF    NB,LP11$K_LEN,  LP11$K_LEN:
                            00000E33              .RESTORE
                31 31 50 4C 00' 0E33              .ASCIC  "LP11"  ; LP11 name
                          04  0E33
                          00  0E38                .BYTE   0                   ; Maximum unit number
                          80  0E39                .BYTE   ^X80                ; Constant to identify start of descriptor
                          83  0E3A                .BYTE   ^X83                ; Indicate scan octal
                   52 53 43 00' 0E3B              .ASCIC  "CSR"   ; CSR string
                          03  0E3B
              0003FFFE 0003E000  0E3F             .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                          83  0E47                .BYTE   ^X83                ; Indicate scan octal
          52 4F 54 43 45 56 00' 0E48              .ASCIC  "VECTOR"            ; VECTOR string
                          06  0E48
              000001FE 00000002  0E4F             .LONG   ^02,^0776           ; 2 , 776 limits on input
                          82  0E57                .BYTE   ^X82                ; Indicate scan decimal
                   52 42 00'  0E58                .ASCIC  "BR"    ; BR string
                          02  0E58
              00000007 00000004  0E5B             .LONG   4,7     ; 4 , 7 limits on input
                          81  0E63                .BYTE   ^X81                ; Indicate end of PT-descriptor
                              0E64    287 LP14:   $DS_LP14
                              0E64                .SAVE   LOCAL_BLOCK
                              0E64                .PSECT  $ABS$,ABS
                      00000032  0046              .=HP$A_DEPENDENT
                                0032              .IIF    NB,LP14$K_LEN,  LP14$K_LEN:
                            00000E64              .RESTORE
                34 31 50 4C 00' 0E64              .ASCIC  "LP14"  ; LP14 name
                          04  0E64
                          01  0E69                .BYTE   1                   ; Maximum unit number
                          80  0E6A                .BYTE   ^X80                ; Constant to identify start of descriptor
                          81  0E6B                .BYTE   ^X81                ; Indicate end of PT-descriptor
                              0E6C    288 LP25:   $DS_LP25                        ;                                       [01]
                              0E6C                .SAVE   LOCAL_BLOCK
                              0E6C                .PSECT  $ABS$,ABS
                      00000032  0046              .=HP$A_DEPENDENT
                                0032              .IIF    NB,LP25$K_LEN,  LP25$K_LEN:
```

```
                        00000E6C                .RESTORE
                35 32 50 4C 00' 0E6C            .ASCIC  "LP25"  ; LP25 name
                        04  0E6C
                        01  0E71                .BYTE   1               ; Maximum unit number
                        80  0E72                .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  0E73                .BYTE   ^X81            ; Indicate end of PT-descriptor
                            0E74    289 LP26:   $DS_LP26                        ;                                   [15]
                            0E74                .SAVE   LOCAL_BLOCK
                            0E74                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,LP26$K_LEN,  LP26$K_LEN:
                        00000E74                .RESTORE
                36 32 50 4C 00' 0E74            .ASCIC  "LP26"  ; LP26 name
                        04  0E74
                        01  0E79                .BYTE   1               ; Maximum unit number
                        80  0E7A                .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  0E7B                .BYTE   ^X81            ; Indicate end of PT-descriptor
                            0E7C    290 LP27:   $DS_LP27                        ;                                   [15]
                            0E7C                .SAVE   LOCAL_BLOCK
                            0E7C                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,LP27$K_LEN,  LP27$K_LEN:
                        00000E7C                .RESTORE
                37 32 50 4C 00' 0E7C            .ASCIC  "LP27"  ; LP27 name
                        04  0E7C
                        01  0E81                .BYTE   1               ; Maximum unit number
                        80  0E82                .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  0E83                .BYTE   ^X81            ; Indicate end of PT-descriptor
                            0E84    291 LPA11K: $DS_LPA11K
                            0E84                .SAVE   LOCAL_BLOCK
                            0E84                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,LPA11K$L_CSR,        LPA11K$L_CSR:
                    00000036  0032              .IIF    NB,.BLKL,       .BLKL   1
                              0036              .IIF    NB,LPA11K$B_BR, LPA11K$B_BR:
                    00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,LPA11K$K_LEN,        LPA11K$K_LEN:
                        00000E84                .RESTORE
             48 31 31 41 50 4C 00' 0E84         .ASCIC  "LPA11K"        ; LPA11K name
                        06  0E84
                        01  0E8B                .BYTE   1               ; Maximum unit number
                        80  0E8C                .BYTE   ^X80            ; Constant to identify start of descriptor
                        83  0E8D                .BYTE   ^X83            ; Indicate scan octal
                52 53 43 00' 0E8E               .ASCIC  "CSR"   ; CSR string
                        03  0E8E
           0003FFFE 0003E000  0E92              .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                        83  0E9A                .BYTE   ^X83            ; Indicate scan octal
             52 4F 54 43 45 56 00' 0E9B         .ASCIC  "VECTOR"        ; VECTOR string
                        06  0E9B
           000001FE 00000002  0EA2              .LONG   ^02,^0776       ; 2 , 776 limits on input
                        82  0EAA                .BYTE   ^X82            ; Indicate scan decimal
                    52 42 00' 0EAB              .ASCIC  "BR"    ; BR string
                        02  0EAB
           00000007 00000004  0EAE              .LONG   4,7     ; 4 , 7 limits on input
                        81  0EB6                .BYTE   ^X81            ; Indicate end of PT-descriptor
                            0EB7    292 MA780:  $DS_MA780
                            0EB7                .SAVE   LOCAL_BLOCK
```

```
                        0EB7                .PSECT  $ABS$,ABS
                00000032 0046               .=HP$A_DEPENDENT
                         0032               .IIF    NB,MA780$B_TR,  MA780$B_TR:
                00000033 0032               .IIF    NB,.BLKB,       .BLKB   1
                         0033               .IIF    NB,MA780$B_BR,  MA780$B_BR:
                00000034 0033               .IIF    NB,.BLKB,       .BLKB   1
                         0034               .IIF    NB,MA780$B_MPM, MA780$B_MPM:
                00000035 0034               .IIF    NB,.BLKB,       .BLKB   1
                         0035               .IIF    NB,MA780$B_PORT,        MA780$B_PORT:
                00000036 0035               .IIF    NB,.BLKB,       .BLKB   1
                         0036               .IIF    NB,MA780$K_LEN, MA780$K_LEN:
                     00000EB7               .RESTORE
      30 38 37 41 4D 00' 0EB7               .ASCIC  "MA780" ; MA780 name
                      05 0EB7
                      08 0EBD               .BYTE   8               ; Maximum unit number
                      80 0EBE               .BYTE   ^X80            ; Constant to identify start of descriptor
                      82 0EBF               .BYTE   ^X82            ; Indicate scan decimal
                52 54 00' 0EC0              .ASCIC  "TR"    ; TR string
                      02 0EC0
      0000000F 00000001 0EC3                .LONG   1,15    ; 1 , 15 limits on input
                      82 0ECB               .BYTE   ^X82            ; Indicate scan decimal
                52 42 00' 0ECC              .ASCIC  "BR"    ; BR string
                      02 0ECC
      00000007 00000004 0ECF                .LONG   4,7     ; 4 , 7 limits on input
                      82 0ED7               .BYTE   ^X82            ; Indicate scan decimal
                4D 50 4D 00' 0ED8           .ASCIC  "MPM"   ; MPM string
                      03 0ED8
      00000003 00000000 0EDC                .LONG   0,3     ; 0 , 3 limits on input
                      82 0EE4               .BYTE   ^X82            ; Indicate scan decimal
             54 52 4F 50 00' 0EE5           .ASCIC  "PORT"  ; PORT string
                      04 0EE5
      00000003 00000000 0EEA                .LONG   0,3     ; 0 , 3 limits on input
                      81 0EF2               .BYTE   ^X81            ; Indicate end of PT-descriptor
                         0EF3   293 MBE:    $DS_MBE
                         0EF3               .SAVE   LOCAL_BLOCK
                         0EF3               .PSECT  $ABS$,ABS
                00000032 0046               .=HP$A_DEPENDENT
                         0032               .IIF    NB,MBE$K_LEN,   MBE$K_LEN:
                     00000EF3               .RESTORE
                45 42 4D 00' 0EF3           .ASCIC  "MBE"   ; MBE name
                      03 0EF3
                      08 0EF7               .BYTE   8               ; Maximum unit number
                      80 0EF8               .BYTE   ^X80            ; Constant to identify start of descriptor
                      81 0EF9               .BYTE   ^X81            ; Indicate end of PT-descriptor
                         0EFA   294 ML11:   $DS_ML11                        ;                                       [01]
                         0EFA               .SAVE   LOCAL_BLOCK
                         0EFA               .PSECT  $ABS$,ABS
                00000032 0046               .=HP$A_DEPENDENT
                         0032               .IIF    NB,ML11$B_ARRAY,        ML11$B_ARRAY:
                00000033 0032               .IIF    NB,.BLKB,       .BLKB   1
                         0033               .IIF    NB,ML11$B_CHIP, ML11$B_CHIP:
                00000034 0033               .IIF    NB,.BLKB,       .BLKB   1
                         0034               .IIF    NB,ML11$K_LEN,  ML11$K_LEN:
                     00000EFA               .RESTORE
             31 31 4C 4D 00' 0EFA           .ASCIC  "ML11"  ; ML11 name
                      04 0EFA
                      08 0EFF               .BYTE   8               ; Maximum unit number
```

```
                                80  0F00                .BYTE   ^X80            ; Constant to identify start of descriptor
                                82  0F01                .BYTE   ^X82            ; Indicate scan decimal
61 20 66 6F 20 72 65 62 6D 75 4E 00' 0F02               .ASCIC  "Number of array boards"        ; Number of array boards string
   73 64 72 61 6F 62 20 79 61 72 72  0F0E
                                16  0F02
                  00000010 00000001  0F19               .LONG   1,16    ; 1 , 16 limits on input
                                85  0F21                .BYTE   ^X85            ; Indicate scan and verify string
      65 7A 69 73 20 70 69 68 43 00' 0F22               .ASCIC  "Chip size"     ; Chip size string
                                09  0F22
                        4B 36 31 00' 0F2C               .ASCIC  "16K"
                                03  0F2C
                        4B 34 36 00' 0F30               .ASCIC  "64K"
                                03  0F30
                                00  0F34               .BYTE   0               ; Null length is end of valid 16K,64K
                                81  0F35               .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0F36    295 MS750:  $DS_MS750
                                    0F36                .SAVE   LOCAL_BLOCK
                                    0F36                .PSECT  $ABS$,ABS
                           00000032 0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,MS750$K_LEN, MS750$K_LEN:
                                00000F36                .RESTORE
                  30 35 37 53 4D 00' 0F36               .ASCIC  "MS750" ; MS750 name
                                05  0F36
                                08  0F3C               .BYTE   8               ; Maximum unit number
                                80  0F3D               .BYTE   ^X80            ; Constant to identify start of descriptor
                                81  0F3E               .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0F3F    296 MS780:  $DS_MS780
                                    0F3F                .SAVE   LOCAL_BLOCK
                                    0F3F                .PSECT  $ABS$,ABS
                           00000032 0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,MS780$B_TR,  MS780$B_TR:
                           00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                                    0033                .IIF    NB,ms780$b_arrays,      ms780$b_arrays:
                           00000034 0033                .IIF    NB,.blkb,       .blkb   1
                                    0034                .IIF    NB,MS780$K_LEN, MS780$K_LEN:
                                00000F3F                .RESTORE
                  30 38 37 53 4D 00' 0F3F               .ASCIC  "MS780" ; MS780 name
                                05  0F3F
                                08  0F45               .BYTE   8               ; Maximum unit number
                                80  0F46               .BYTE   ^X80            ; Constant to identify start of descriptor
                                82  0F47               .BYTE   ^X82            ; Indicate scan decimal
                        52 54 00' 0F48               .ASCIC  "TR"    ; TR string
                                02  0F48
                  0000000F 00000001  0F48               .LONG   1,15    ; 1 , 15 limits on input
                                82  0F53               .BYTE   ^X82            ; Indicate scan decimal
61 20 66 6F 20 72 65 62 6D 75 4E 00' 0F54               .ASCIC  "Number of arrays"      ; Number of arrays string
                   73 79 61 72 72  0F60
                                10  0F54
                  00000010 00000000  0F65               .LONG   0,16    ; 0 , 16 limits on input
                                81  0F6D               .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    0F6E    297 PCL11:  $DS_PCL11
                                    0F6E                .SAVE   LOCAL_BLOCK
                                    0F6E                .PSECT  $ABS$,ABS
                           00000032 0046                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,PCL11$L_CSR, PCL11$L_CSR:
                           00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                                    0036                .IIF    NB,PCL11$B_BR,  PCL11$B_BR:
```

```
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,PCL11$K_LEN, PCL11$K_LEN:
                00000F6E                      .RESTORE
      31 31 4C 43 50 00'  0F6E                .ASCIC  "PCL11" ; PCL11 name
                      05  0F6E
                      20  0F74                .BYTE   32              ; Maximum unit number
                      80  0F75                .BYTE   ^X80            ; Constant to identify start of descriptor
                      83  0F76                .BYTE   ^X83            ; Indicate scan octal
            52 53 43 00'  0F77                .ASCIC  "CSR"   ; CSR string
                      03  0F77
      0003FFFE 0003E000   0F78                .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                      83  0F83                .BYTE   ^X83            ; Indicate scan octal
   52 4F 54 43 45 56 00'  0F84                .ASCIC  "VECTOR"        ; VECTOR string
                      06  0F84
      000001FE 00000002   CF8B                .LONG   ^02,^0776       ; 2 , 776 limits on input
                      82  0F93                .BYTE   ^X82            ; Indicate scan decimal
                52 42 00'  0F94                .ASCIC  "BR"    ; BR string
                      02  0F94
      00000007 00000004   0F97                .LONG   4,7     ; 4 , 7 limits on input
                      81  0F9F                .BYTE   ^X81            ; Indicate end of PT-descriptor
                          0FA0    298 PX780:  $DS_PX780                       ;
                          0FA0                .SAVE   LOCAL_BLOCK
                          0FA0                .PSECT  $ABS$,ABS
                00000032  0046                .=HP$A_DEPENDENT
                          0032                .IIF    NB,PX780$B_TR,  PX780$B_TR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,PX780$B_BR,  PX780$B_BR:
                00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,PX780$B_NODE,        PX780$B_NODE:
                000J0035  0034                .IIF    NB,.BLKB,       .BLKB   1
                          0035                .IIF    NB,PX780$K_LEN, PX780$K_LEN:
                00000FA0                      .RESTORE
   30 38 37 58 50 00'  0FA0                .ASCIC  "PX780" ; PX780 name
                      05  0FA0
                      01  0FA6                .BYTE   1               ; Maximum unit number
                      80  0FA7                .BYTE   ^X80            ; Constant to identify start of descriptor
                      82  0FA8                .BYTE   ^X82            ; Indicate scan decimal
                52 54 00'  0FA9                .ASCIC  "TR"    ; TR string
                      02  0FA9
      0000000F 00000001   0FAC                .LONG   1,15    ; 1 , 15 limits on input
                      82  0FB4                .BYTE   ^X82            ; Indicate scan decimal
                52 42 00'  0FB5                .ASCIC  "BR"    ; BR string
                      02  0FB5
      00000007 00000004   0FB8                .LONG   4,7     ; 4 , 7 limits on input
                      82  0FC0                .BYTE   ^X82            ; Indicate scan decimal
         65 64 6F 4E 00'  0FC1                .ASCIC  "Node"  ; Node string
                      04  0FC1
      000000FF 00000000   0FC6                .LONG   0,255   ; 0 , 255 limits on input
                      81  0FCE                .BYTE   ^X81            ; Indicate end of PT-descriptor
                          0FCF    299 R80:    $DS_R80                             ;                                   [02]
                          0FCF                .SAVE   LOCAL_BLOCK
                          0FCF                .PSECT  $ABS$,ABS
                00000032  0046                .=HP$A_DEPENDENT
                          0032                .IIF    NB,R80$K_LEN,   R80$K_LEN:
                00000FCF                      .RESTORE
            30 38 52 00'  0FCF                .ASCIC  "R80"   ; R80 name
                      03  0FCF
```

```
                        08  0FD3                .BYTE   8                   ; Maximum unit number
                        80  0FD4                .BYTE   ^X80                ; Constant to identify start of descriptor
                        81  0FD5                .BYTE   ^X81                ; Indicate end of PT-descriptor
                            0FD6    300 RA60:   $DS_RA60
                            0FD6                .SAVE   LOCAL_BLOCK
                            0FD6                .PSECT  $ABS$,ABS
                   00000032 0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_RA60_LEN,       HP$K_RA60_LEN:
                            0032                .IIF    NB,RA60$K_LEN,  RA60$K_LEN:
                        00000FD6                .RESTORE
                30 36 41 52 00' 0FD6            .ASCIC  "RA60"  ; RA60 name
                        04  0FD6
                        FF  0FDB                .BYTE   255                 ; Maximum unit number
                        80  0FDC                .BYTE   ^X80                ; Constant to identify start of descriptor
                        81  0FDD                .BYTE   ^X81                ; Indicate end of PT-descriptor
                            0FDE    301 RA80:   $DS_RA80                                ;                                   [02]
                            0FDE                .SAVE   LOCAL_BLOCK
                            0FDE                .PSECT  $ABS$,ABS
                   00000032 0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_RA80_LEN,       HP$K_RA80_LEN:
                            0032                .IIF    NB,RA80$K_LEN,  RA80$K_LEN:
                        00000FDE                .RESTORE
                30 38 41 52 00' 0FDE            .ASCIC  "RA80"  ; RA80 name
                        04  0FDE
                        FF  0FE3                .BYTE   255                 ; Maximum unit number
                        80  0FE4                .BYTE   ^X80                ; Constant to identify start of descriptor
                        81  0FE5                .BYTE   ^X81                ; Indicate end of PT-descriptor
                            0FE6    302 RA81:   $DS_RA81
                            0FE6                .SAVE   LOCAL_BLOCK
                            0FE6                .PSECT  $ABS$,ABS
                   00000032 0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_RA81_LEN,       HP$K_RA81_LEN:
                            0032                .IIF    NB,RA81$K_LEN,  RA81$K_LEN:
                        00000FE6                .RESTORE
                31 38 41 52 00' 0FE6            .ASCIC  "RA81"  ; RA81 name
                        04  0FE6
                        FF  0FEB                .BYTE   255                 ; Maximum unit number
                        80  0FEC                .BYTE   ^X80                ; Constant to identify start of descriptor
                        81  0FED                .BYTE   ^X81                ; Indicate end of PT-descriptor
                            0FEE    303 RB730:  $DS_RB730                               ;                                   [02]
                            0FEE                .SAVE   LOCAL_BLOCK
                            0FEE                .PSECT  $ABS$,ABS
                   00000032 0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$L_RB730_CSR,      HP$L_RB730_CSR:
                            0032                .IIF    NB,RB730$L_CSR, RB730$L_CSR:
                   00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                            0036                .IIF    NB,HP$B_RB730_BR,       HP$B_RB730_BR:
                            0036                .IIF    NB,RB730$B_BR,  RB730$B_BR:
                   00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                            0037                .IIF    NB,HP$K_RB730_LEN,      HP$K_RB730_LEN:
                            0037                .IIF    NB,RB730$K_LEN, RB730$K_LEN:
                        00000FEE                .RESTORE
             30 33 37 42 52 00' 0FEE            .ASCIC  "RB730" ; RB730 name
                        05  0FEE
                        00  0FF4                .BYTE   0                   ; Maximum unit number
                        80  0FF5                .BYTE   ^X80                ; Constant to identify start of descriptor
                        81  0FF6                .BYTE   ^X81                ; Indicate end of PT-descriptor
```

```
                        0FF7   304 RCF25:  $DS_RCF25                                  ;                                  [12]
                        0FF7               .SAVE   LOCAL_BLOCK
                        0FF7               .PSECT  $ABS$,ABS
                00000032  0046             .=HP$A_DEPENDENT
                          0032             .IIF    NB,HP$K_RCF25_LEN,      HP$K_RCF25_LEN:
                          0032             .IIF    NB,RCF25$K_LEN, RCF25$K_LEN:
                        0FF7               .RESTORE
          35 32 46 43 52 00'  0FF7         .ASCIC  "RCF25" ; RCF25 name
                          05  0FF7
                          FF  0FFD         .BYTE   255             ; Maximum unit number
                          80  0FFE         .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  0FFF         .BYTE   ^X81            ; Indicate end of PT-descriptor
                        1000   305 ;[22] RC11:     $DS_RC11                        ;                                  [15]
                        1000   306 RC25:   $DS_RC25                                ;                                  [11]
                        1000               .SAVE   LOCAL_BLOCK
                        1000               .PSECT  $ABS$,ABS
                00000032  0046             .=HP$A_DEPENDENT
                          0032             .IIF    NB,HP$K_RC25_LEN,       HP$K_RC25_LEN:
                          0032             .IIF    NB,RC25$K_LEN,  RC25$K_LEN:
                        1000               .RESTORE
             35 32 43 52 00'  1000         .ASCIC  "RC25"  ; RC25 name
                          04  1000
                          FF  1005         .BYTE   255             ; Maximum unit number
                          80  1006         .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  1007         .BYTE   ^X81            ; Indicate end of PT-descriptor
                        1008   307 RD51:   $DS_RD51                                  ;                                  [19]
                        1008               .SAVE   LOCAL_BLOCK
                        1008               .PSECT  $ABS$,ABS
                00000032  0046             .=HP$A_DEPENDENT
                          0032             .IIF    NB,RD51$K_LEN,  RD51$K_LEN:
                        1008               .RESTORE
          31 35 44 52 00'  1008            .ASCIC  "RD51"  ; RD51 name
                          04  1008
                          FF  100D         .BYTE   255             ; Maximum unit number
                          80  100E         .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  100F         .BYTE   ^X81            ; Indicate end of PT-descriptor
                        1010   308 RD52:   $DS_RD52                                  ;                                  [19]
                        1010               .SAVE   LOCAL_BLOCK
                        1010               .PSECT  $ABS$,ABS
                00000032  0046             .=HP$A_DEPENDENT
                          0032             .IIF    NB,RD52$K_LEN,  RD52$K_LEN:
                        1010               .RESTORE
          32 35 44 52 00'  1010            .ASCIC  "RD52"  ; RD52 name
                          04  1010
                          FF  1015         .BYTE   255             ; Maximum unit number
                          80  1016         .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  1017         .BYTE   ^X81            ; Indicate end of PT-descriptor
                        1018   309 RH750:   $DS_RH750
                        1018               .SAVE   LOCAL_BLOCK
                        1018               .PSECT  $ABS$,ABS
                00000032  0046             .=HP$A_DEPENDENT
                          0032             .IIF    NB,HP$B_RH750_BR,       HP$B_RH750_BR:
                          0032             .IIF    NB,RH750$B_BR,  RH750$B_BR:
                00000033  0032             .IIF    NB,.BLKB,       .BLKB   1
                          0033             .IIF    NB,HP$K_RH750_LEN,      HP$K_RH750_LEN:
                          0033             .IIF    NB,RH750$K_LEN, RH750$K_LEN:
                        1018               .RESTORE
```

```
                30 35 37 48 52 00' 1018                 .ASCIC  "RH750" ; RH750 name
                                05  1018
                                08  101E                 .BYTE   8               ; Maximum unit number
                                80  101F                 .BYTE   ^X80            ; Constant to identify start of descriptor
                                82  1020                 .BYTE   ^X82            ; Indicate scan decimal
                          52 42 00' 1021                 .ASCIC  "BR"    ; BR string
                                02  1021
                00000007 00000004   1024                 .LONG   4,7     ; 4 , 7 limits on input
                                81  102C                 .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    102D    310 RH780:   $DS_RH780
                                    102D                 .SAVE   LOCAL_BLOCK
                                    102D                 .PSECT  $ABS$,ABS
                          00000032  0046                 .=HP$A_DEPENDENT
                                    0032                 .IIF    NB,HP$B_RH780_TR,       HP$B_RH780_TR:
                                    C032                 .IIF    NB,RH780$B_TR,  RH780$B_TR:
                          00000033  0032                 .IIF    NB,.BLKB,       .BLKB   1
                                    0033                 .IIF    NB,HP$B_RH780_BR,       HP$B_RH780_BR:
                                    0033                 .IIF    NB,RH780$B_BR,  RH780$B_BR:
                          00000034  0033                 .IIF    NB,.BLKB,       .BLKB   1
                                    0034                 .IIF    NB,HP$K_RH780_LEN,      HP$K_RH780_LEN:
                                    0034                 .IIF    NB,RH780$K_LEN, RH780$K_LEN:
                                0000102D                 .RESTORE
                30 38 37 48 52 00' 102D                 .ASCIC  "RH780" ; RH780 name
                                05  102D
                                08  1033                 .BYTE   8               ; Maximum unit number
                                80  1034                 .BYTE   ^X80            ; Constant to identify start of descriptor
                                82  1035                 .BYTE   ^X82            ; Indicate scan decimal
                          52 54 00' 1036                 .ASCIC  "TR"    ; TR string
                                02  1036
                0000000F 00000001   1039                 .LONG   1,15    ; 1 , 15 limits on input
                                82  1041                 .BYTE   ^X82            ; Indicate scan decimal
                          52 42 00' 1042                 .ASCIC  "BR"    ; BR string
                                02  1042
                00000007 00000004   1045                 .LONG   4,7     ; 4 , 7 limits on input
                                81  104D                 .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    104E    311 RL01:    $DS_RL01
                                    104E                 .SAVE   LOCAL_BLOCK
                                    104E                 .PSECT  $ABS$,ABS
                          00000032  0046                 .=HP$A_DEPENDENT
                                    0032                 .IIF    NB,HP$K_RL01_LEN,       HP$K_RL01_LEN:
                                    0032                 .IIF    NB,RL01$K_LEN,  RL01$K_LEN:
                                0000104E                 .RESTORE
                   31 30 4C 52 00' 104E                 .ASCIC  "RL01"  ; RL01 name
                                04  104E
                                08  1053                 .BYTE   8               ; Maximum unit number
                                80  1054                 .BYTE   ^X80            ; Constant to identify start of descriptor
                                81  1055                 .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    1056    312 RL02:    $DS_RL02
                                    1056                 .SAVE   LOCAL_BLOCK
                                    1056                 .PSECT  $ABS$,ABS
                          00000032  0046                 .=HP$A_DEPENDENT
                                    0032                 .IIF    NB,HP$K_RL02_LEN,       HP$K_RL02_LEN:
                                    0032                 .IIF    NB,RL02$K_LEN,  RL02$K_LEN:
                                00001056                 .RESTORE
                   32 30 4C 52 00' 1056                 .ASCIC  "RL02"  ; RL02 name
                                04  1056
                                08  105B                 .BYTE   8               ; Maximum unit number
```

```
                          80  105C                .BYTE   ^X80              ; Constant to identify start of descriptor
                          81  105D                .BYTE   ^X81              ; Indicate end of PT-descriptor
                              105E    313 RL11:   $DS_RL11
                              105E                .SAVE   LOCAL_BLOCK
                              105E                .PSECT  $ABS$,ABS
                      00000032  0046                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$L_RL11_CSR,       HP$L_RL11_CSR:
                              0032                .IIF    NB,RL11$L_CSR,  RL11$L_CSR:
                      00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                              0036                .IIF    NB,HP$B_RL11_BR,        HP$B_RL11_BR:
                              0036                .IIF    NB,RL11$B_BR,   RL11$B_BR:
                      00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                              0037                .IIF    NB,HP$K_RL11_LEN,       HP$K_RL11_LEN:
                              0037                .IIF    NB,RL11$K_LEN,  RL11$K_LEN:
                          0000105E                .RESTORE
                  31 31 4C 52 00' 105E            .ASCIC  "RL11"  ; RL11 name
                          04  105E
                          00  1063                .BYTE   0               ; Maximum unit number
                          80  1064                .BYTE   ^X80            ; Constant to identify start of descriptor
                          83  1065                .BYTE   ^X83            ; Indicate scan octal
                      52 53 43 00' 1066           .ASCIC  "CSR"   ; CSR string
                          03  1066
                  0003FFFE 0003E000  106A         .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                          83  1072                .BYTE   ^X83            ; Indicate scan octal
                52 4F 54 43 45 56 00' 1073        .ASCIC  "VECTOR"        ; VECTOR string
                          06  1073
                  000001FE 00000002  107A         .LONG   ^02,^0776       ; 2 , 776 limits on input
                          82  1082                .BYTE   ^X82            ; Indicate scan decimal
                      52 42 00' 1083              .ASCIC  "BR"    ; BR string
                          02  1083
                  00000007 00000004  1086         .LONG   4,7     ; 4 , 7 limits on input
                          81  108E                .BYTE   ^X81            ; Indicate end of PT-descriptor
                              108F    314 RK06:   $DS_RK06
                              108F                .SAVE   LOCAL_BLOCK
                              108F                .PSECT  $ABS$,ABS
                      00000032  0046                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$K_RK06_LEN,       HP$K_RK06_LEN:
                              0032                .IIF    NB,RK06$K_LEN,  RK06$K_LEN:
                          0000108F                .RESTORE
                  36 30 4B 52 00' 108F            .ASCIC  "RK06"  ; RK06 name
                          04  108F
                          08  1094                .BYTE   8               ; Maximum unit number
                          80  1095                .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  1096                .BYTE   ^X81            ; Indicate end of PT-descriptor
                              1097    315 RK07:   $DS_RK07
                              1097                .SAVE   LOCAL_BLOCK
                              1097                .PSECT  $ABS$,ABS
                      00000032  0046                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$K_RK07_LEN,       HP$K_RK07_LEN:
                              0032                .IIF    NB,RK07$K_LEN,  RK07$K_LEN:
                          00001097                .RESTORE
                  37 30 4B 52 00' 1097            .ASCIC  "RK07"  ; RK07 name
                          04  1097
                          08  109C                .BYTE   8               ; Maximum unit number
                          80  109D                .BYTE   ^X80            ; Constant to identify start of descriptor
                          81  109E                .BYTE   ^X81            ; Indicate end of PT-descriptor
                              109F    316 RK611:  $DS_RK611
```

```
                                109F                .SAVE   LOCAL_BLOCK
                                109F                .PSECT  $ABS$,ABS
                    00000032    0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$L_RK611_CSR,      HP$L_RK611_CSR:
                                0032                .IIF    NB,RK611$L_CSR, RK611$L_CSR:
                    00000036    0032                .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,HP$B_RK611_BR,       HP$B_RK611_BR:
                                0036                .IIF    NB,RK611$B_BR,  RK611$B_BR:
                    00000037    0036                .IIF    NB,.BLKB,       .BLKB   1
                                0037                .IIF    NB,HP$K_RK611_LEN,      HP$K_RK611_LEN:
                                0037                .IIF    NB,RK611$K_LEN, RK611$K_LEN:
                              0000109F              .RESTORE
                 31 31 36 4B 52 00' 109F            .ASCIC  "RK611" ; RK611 name
                                05  109F
                                00  10A5            .BYTE   0               ; Maximum unit number
                                80  10A6            .BYTE   ^X80            ; Constant to identify start of descriptor
                                83  10A7            .BYTE   ^X83            ; Indicate scan octal
                          52 53 43 00' 10A8         .ASCIC  "CSR"   ; CSR string
                                03  10A8
                    0003FFFE 0003E000   10AC        .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                                83  10B4            .BYTE   ^X83            ; Indicate scan octal
                 52 4F 54 43 45 56 00' 10B5         .ASCIC  "VECTOR"        ; VECTOR string
                                06  10B5
                    000001FE 00000002   10BC        .LONG   ^O2,^0776       ; 2 , 776 limits on input
                                82  10C4            .BYTE   ^X82            ; Indicate scan decimal
                          52 42 00' 10C5            .ASCIC  "BR"    ; BR string
                                02  10C5
                    00000007 00000004   10C8        .LONG   4,7     ; 4 , 7 limits on input
                                81  10D0            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    10D1    317 RM03:   $DS_RM03
                                10D1                .SAVE   LOCAL_BLOCK
                                10D1                .PSECT  $ABS$,ABS
                          00000032  0046            .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$K_RM03_LEN,       HP$K_RM03_LEN:
                                0032                .IIF    NB,RM03$K_LEN,  RM03$K_LEN:
                              000010D1              .RESTORE
                       33 30 4D 52 00' 10D1         .ASCIC  "RM03"  ; RM03 name
                                04  10D1
                                08  10D6            .BYTE   8               ; Maximum unit number
                                80  10D7            .BYTE   ^X80            ; Constant to identify start of descriptor
                                81  10D8            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    10D9    318 RM05:   $DS_RM05
                                10D9                .SAVE   LOCAL_BLOCK
                                10D9                .PSECT  $ABS$,ABS
                          00000032  0046            .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$K_RM05_LEN,       HP$K_RM05_LEN:
                                0032                .IIF    NB,RM05$K_LEN,  RM05$K_LEN:
                              000010D9              .RESTORE
                       35 30 4D 52 00' 10D9         .ASCIC  "RM05"  ; RM05 name
                                04  10D9
                                08  10DE            .BYTE   8               ; Maximum unit number
                                80  10DF            .BYTE   ^X80            ; Constant to identify start of descriptor
                                81  10E0            .BYTE   ^X81            ; Indicate end of PT-descriptor
                                    10E1    319 RM80:   $DS_RM80
                                10E1                .SAVE   LOCAL_BLOCK
                                10E1                .PSECT  $ABS$,ABS
                          00000032  0046            .=HP$A_DEPENDENT
```

```
                        0032                .IIF    NB,HP$K_RM80_LEN,       HP$K_RM80_LEN:
                        0032                .IIF    NB,RM80$K_LEN,  RM80$K_LEN:
                    000010E1                .RESTORE
          30 38 4D 52 00' 10E1              .ASCIC  "RM80"  ; RM80 name
                        04  10E1
                        08  10E6              .BYTE   8               ; Maximum unit number
                        80  10E7              .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  10E8              .BYTE   ^X81            ; Indicate end of PT-descriptor
                            10E9    320 RP04:   $DS_RP04
                            10E9              .SAVE   LOCAL_BLOCK
                            10E9              .PSECT  $ABS$,ABS
                    00000032  0046            .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_RP04_LEN,       HP$K_RP04_LEN:
                        0032                .IIF    NB,RP04$K_LEN,  RP04$K_LEN:
                    000010E9                .RESTORE
          34 30 50 52 00' 10E9              .ASCIC  "RP04"  ; RP04 name
                        04  10E9
                        08  10EE              .BYTE   8               ; Maximum unit number
                        80  10EF              .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  10F0              .BYTE   ^X81            ; Indicate end of PT-descriptor
                            10F1    321 RP05:   $DS_RP05
                            10F1              .SAVE   LOCAL_BLOCK
                            10F1              .PSECT  $ABS$,ABS
                    00000032  0046            .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_RP05_LEN,       HP$K_RP05_LEN:
                        0032                .IIF    NB,RP05$K_LEN,  RP05$K_LEN:
                    000010F1                .RESTORE
          35 30 50 52 00' 10F1              .ASCIC  "RP05"  ; RP05 name
                        04  10F1
                        08  10F6              .BYTE   8               ; Maximum unit number
                        80  10F7              .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  10F8              .BYTE   ^X81            ; Indicate end of PT-descriptor
                            10F9    322 RP06:   $DS_RP06
                            10F9              .SAVE   LOCAL_BLOCK
                            10F9              .PSECT  $ABS$,ABS
                    00000032  0046            .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_RP06_LEN,       HP$K_RP06_LEN:
                        0032                .IIF    NB,RP06$K_LEN,  RP06$K_LEN:
                    000010F9                .RESTORE
          36 30 50 52 00' 10F9              .ASCIC  "RP06"  ; RP06 name
                        04  10F9
                        08  10FE              .BYTE   8               ; Maximum unit number
                        80  10FF              .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  1100              .BYTE   ^X81            ; Indicate end of PT-descriptor
                            1101    323 RP07:   $DS_RP07
                            1101              .SAVE   LOCAL_BLOCK
                            1101              .PSECT  $ABS$,ABS
                    00000032  0046            .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_RP07_LEN,       HP$K_RP07_LEN:
                        0032                .IIF    NB,RP07$K_LEN,  RP07$K_LEN:
                    00001101                .RESTORE
          37 30 50 52 00' 1101              .ASCIC  "RP07"  ; RP07 name
                        04  1101
                        08  1106              .BYTE   8               ; Maximum unit number
                        80  1107              .BYTE   ^X80            ; Constant to identify start of descriptor
                        81  1108              .BYTE   ^X81            ; Indicate end of PT-descriptor
                            1109    324 RQDX1:  $DS_RQDX1                       ;                                   [19]
```

```
                                1109                .SAVE   LOCAL_BLOCK
                                1109                .PSECT  $ABS$,ABS
                      00000032  0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,RQDX1$L_IP,  RQDX1$L_IP:
                      00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,RQDX1$K_LEN, RQDX1$K_LEN:
                            0000 1109               .RESTORE
                31 58 44 51 52 00' 1109             .ASCIC  "RQDX1" ; RQDX1 name
                            05  1109
                            00  110F                .BYTE   0               ; Maximum unit number
                            80  1110                .BYTE   ^X80            ; Constant to identify start of descriptor
                            83  1111                .BYTE   ^X83            ; Indicate scan octal
                      50 49 00' 1112                .ASCIC  "IP"    ; IP string
                            02  1112
                0003FFFC 0003E000 1115              .LONG   ^0760000,^0777774       ; 760000 , 777774 limits on input
                            81  111D                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                111E    325 RX02:   $DS_RX02
                                111E                .SAVE   LOCAL_BLOCK
                                111E                .PSECT  $ABS$,ABS
                      00000032  0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$K_RX02_LEN,       HP$K_RX02_LEN:
                                0032                .IIF    NB,RX02$K_LEN,  RX02$K_LEN:
                            0000 111E               .RESTORE
                   32 30 58 52 00' 111E             .ASCIC  "RX02"  ; RX02 name
                            04  111E
                            02  1123                .BYTE   2               ; Maximum unit number
                            80  1124                .BYTE   ^X80            ; Constant to identify start of descriptor
                            81  1125                .BYTE   ^X81            ; Indicate end of PT-descriptor
                                1126    326 RX211:  $DS_RX211
                                1126                .SAVE   LOCAL_BLOCK
                                1126                .PSECT  $ABS$,ABS
                      00000032  0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$L_RX211_CSR,      HP$L_RX211_CSR:
                                0032                .IIF    NB,RX211$L_CSR, RX211$L_CSR:
                      00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,HP$B_RX211_BR,       HP$B_RX211_BR:
                                0036                .IIF    NB,RX211$B_BR,  RX211$B_BR:
                      00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                0037                .IIF    NB,HP$K_RX211_LEN,      HP$K_RX211_LEN:
                                0037                .IIF    NB,RX211$K_LEN, RX211$K_LEN:
                            0000 1126               .RESTORE
                31 31 32 58 52 00' 1126             .ASCIC  "RX211" ; RX211 name
                            05  1126
                            00  112C                .BYTE   0               ; Maximum unit number
                            80  112D                .BYTE   ^X80            ; Constant to identify start of descriptor
                            83  112E                .BYTE   ^X83            ; Indicate scan octal
                      52 53 43 00' 112F             .ASCIC  "CSR"   ; CSR string
                            03  1132
                0003FFFE 0003E000 1133              .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                            83  1138                .BYTE   ^X83            ; Indicate scan octal
             52 4F 54 43 45 56 00' 113C             .ASCIC  "VECTOR"        ; VECTOR string
                            06  113C
                000001FE 00000002 1143              .LONG   ^02,^0776       ; 2 , 776 limits on input
                            82  114B                .BYTE   ^X82            ; Indicate scan decimal
                      52 42 00' 114C                .ASCIC  "BR"    ; BR string
                            02  114C
                00000007 00000004 114F              .LONG   4,7     ; 4 , 7 limits on input
```

```
                      81  1157                .BYTE   ^X81               ; Indicate end of PT-descriptor
                          1158    327 RX50:   $DS_RX50                                                    [19]
                          1158                .SAVE   LOCAL_BLOCK
                          1158                .PSECT  $ABS$,ABS
                00000032  0046                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RX50$K_LEN,  RX50$K_LEN:
                00001158                      .RESTORE
          30 35 58 52 00' 1158                .ASCIC  "RX50"  ; RX50 name
                      04  1158
                      FF  115D                .BYTE   255               ; Maximum unit number
                      80  115E                .BYTE   ^X80              ; Constant to identify start of descriptor
                      81  115F                .BYTE   ^X81              ; Indicate end of PT-descriptor
                          1160    328 SBIA:   $DS_SBIA                                                    [15]
                          1160                .SAVE   LOCAL_BLOCK
                          1160                .PSECT  $ABS$,ABS
                00000032  0046                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_SBIA_LEN,       HP$K_SBIA_LEN:
                          0032                .IIF    NB,SBIA$K_LEN,  SBIA$K_LEN:
                00001160                      .RESTORE
          41 49 42 53 00' 1160                .ASCIC  "SBIA"  ; SBIA name
                      04  1160
                      02  1165                .BYTE   2                 ; Maximum unit number
                      80  1166                .BYTE   ^X80              ; Constant to identify start of descriptor
                      81  1167                .BYTE   ^X81              ; Indicate end of PT-descriptor
                          1168    329 TE16:   $DS_TE16
                          1168                .SAVE   LOCAL_BLOCK
                          1168                .PSECT  $ABS$,ABS
                00000032  0046                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_TE16_LEN,       HP$K_TE16_LEN:
                          0032                .IIF    NB,TE16$K_LEN,  TE16$K_LEN:
                00001168                      .RESTORE
          36 31 45 54 00' 1168                .ASCIC  "TE16"  ; TE16 name
                      04  1168
                      08  116D                .BYTE   8                 ; Maximum unit number
                      80  116E                .BYTE   ^X80              ; Constant to identify start of descriptor
                      81  116F                .BYTE   ^X81              ; Indicate end of PT-descriptor
                          1170    330 TM03:   $DS_TM03
                          1170                .SAVE   LOCAL_BLOCK
                          1170                .PSECT  $ABS$,ABS
                00000032  0046                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_TM03_DRIVE,     HP$B_TM03_DRIVE:
                          0032                .IIF    NB,TM03$B_DRIVE,        TM03$B_DRIVE:
                00000033  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_TM03_LEN,       HP$K_TM03_LEN:
                          0033                .IIF    NB,TM03$K_LEN,  TM03$K_LEN:
                00001170                      .RESTORE
          33 30 4D 54 00' 1170                .ASCIC  "TM03"  ; TM03 name
                      04  1170
                      00  1175                .BYTE   0                 ; Maximum unit number
                      80  1176                .BYTE   ^X80              ; Constant to identify start of descriptor
                      82  1177                .BYTE   ^X82              ; Indicate scan decimal
       45 56 49 52 44 00' 1178                .ASCIC  "DRIVE" ; DRIVE string
                      05  1178
       00000007 00000000  117E                .LONG   0,7     ; 0 , 7 limits on input
                      81  1186                .BYTE   ^X81              ; Indicate end of PT-descriptor
                          1187    331 TM78:   $DS_TM78                                                    [01]
                          1187                .SAVE   LOCAL_BLOCK
```

```
                                1187                .PSECT  $ABS$,ABS
                       00000032 0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$B_TM78_DRIVE,     HP$B_TM78_DRIVE:
                                0032                .IIF    NB,TM78$B_DRIVE,        TM78$B_DRIVE:
                       00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                                0033                .IIF    NB,HP$K_TM78_LEN,       HP$K_TM78_LEN:
                                0033                .IIF    NB,TM78$K_LEN,  TM78$K_LEN:
                            0000 1187                .RESTORE
                  38 37 4D 54 00' 1187              .ASCIC  "TM78"  ; TM78 name
                              04  1187
                              00  118C              .BYTE   0               ; Maximum unit number
                              80  118D              .BYTE   ^X80            ; Constant to identify start of descriptor
                              82  118E              .BYTE   ^X82            ; Indicate scan decimal
               45 56 49 52 44 00' 118F              .ASCIC  "DRIVE" ; DRIVE string
                              05  118F
                00000007 00000000 1195              .LONG   0,7     ; 0 , 7 limits on input
                              81  119D              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                  119E    332 TS04:   $DS_TS04
                                  119E                .SAVE   LOCAL_BLOCK
                                  119E                .PSECT  $ABS$,ABS
                       00000032 0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,HP$L_TS04_CSR,       HP$L_TS04_CSR:
                                0032                .IIF    NB,TS04$L_CSR,  TS04$L_CSR:
                       00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,HP$B_TS04_BR,        HP$B_TS04_BR:
                                0036                .IIF    NB,TS04$B_BR,   TS04$B_BR:
                       00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                                0037                .IIF    NB,HP$K_TS04_LEN,       HP$K_TS04_LEN:
                                0037                .IIF    NB,TS04$K_LEN,  TS04$K_LEN:
                            C000 119E                .RESTORE
                  34 30 53 54 00' 119E              .ASCIC  "TS04"  ; TS04 name
                              04  119E
                              01  11A3              .BYTE   1               ; Maximum unit number
                              80  11A4              .BYTE   ^X80            ; Constant to identify start of descriptor
                              83  11A5              .BYTE   ^X83            ; Indicate scan octal
                     52 53 43 00' 11A6              .ASCIC  "CSR"   ; CSR string
                              03  11A6
                    0003FFFE 0003E000 11AA          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                              83  11B2              .BYTE   ^X83            ; Indicate scan octal
                52 4F 54 43 45 56 00' 11B3          .ASCIC  "VECTOR"        ; VECTOR string
                              06  11B3
                    000001FE 00000002 11BA          .LONG   ^O2,^O776       ; 2 , 776 limits on input
                              82  11C2              .BYTE   ^X82            ; Indicate scan decimal
                        52 42 00' 11C3              .ASCIC  "BR"    ; BR string
                              02  11C3
                00000007 00000004 11C6              .LONG   4,7     ; 4 , 7 limits on input
                              81  11CE              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                  11CF    333 TS05:   $DS_TS05                        ;                                   [18]
                                  11CF                .SAVE   LOCAL_BLOCK
                                  11CF                .PSECT  $ABS$,ABS
                       00000032 0046                .=HP$A_DEPENDENT
                                0032                .IIF    NB,TS05$B_BR,   TS05$B_BR:
                       00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                                0033                .IIF    NB,HP$K_TS05_LEN,       HP$K_TS05_LEN:
                                0033                .IIF    NB,TS05$K_LEN,  TS05$K_LEN:
                            0000 11CF                .RESTORE
                  35 30 53 54 00' 11CF              .ASCIC  "TS05"  ; TS05 name
```

```
                                04  11CF
                                08  11D4                .BYTE   8                       ; Maximum unit number
                                80  11D5                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                83  11D6                .BYTE   ^X83                    ; Indicate scan octal
44 41 5F 52 53 43 5F 35 30 53 54 00'  11D7                .ASCIC  "TSO5_CSR_ADDRESS"      ; TSO5_CSR_ADDRESS string
                     53 53 45 52 44  11E3
                                10  11D7
                  0003FFFE 0003E000  11E8                .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                                83  11F0                .BYTE   ^X83                    ; Indicate scan octal
52 4F 54 43 45 56 5F 35 30 53 54 00'  11F1                .ASCIC  "TSO5_VECTOR"   ; TSO5_VECTOR string
                                08  11F1
                  000001FE 00000002  11FD                .LONG   ^02,^0776       ; 2 , 776 limits on input
                                82  1205                .BYTE   ^X82                    ; Indicate scan decimal
        4C 45 56 45 4C 5F 52 42 00'  1206                .ASCIC  "BR_LEVEL"      ; BR_LEVEL string
                                08  1206
                  00000007 00000004  120F                .LONG   4,7     ; 4 , 7 limits on input
                                81  1217                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                    1218    334 TS11:   $DS_TS11
                                    1218                .SAVE   LOCAL_BLOCK
                                    1218                .PSECT  $ABS$,ABS
                         00000032  0046                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,HP$L_TS11_CSR,       HP$L_TS11_CSR:
                                   0032                .IIF    NB,TS11$L_CSR,  TS11$L_CSR:
                         00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                   0036                .IIF    NB,HP$B_TS11_BR,        HP$B_TS11_BR:
                                   0036                .IIF    NB,TS11$B_BR,   TS11$B_BR:
                         00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                   0037                .IIF    NB,HP$K_TS11_LEN,       HP$K_TS11_LEN:
                                   0037                .IIF    NB,TS11$K_LEN,  TS11$K_LEN:
                              00001218                 .RESTORE
                     31 31 53 54 00'  1218                .ASCIC  "TS11"  ; TS11 name
                                04  1218
                                01  121D                .BYTE   1                       ; Maximum unit number
                                80  121E                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                83  121F                .BYTE   ^X83                    ; Indicate scan octal
                        52 53 43 00'  1220                .ASCIC  "CSR"   ; CSR string
                                03  1220
                     0003FFFE 0003E000  1224                .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                                83  122C                .BYTE   ^X83                    ; Indicate scan octal
                  52 4F 54 43 45 56 00'  122D                .ASCIC  "VECTOR"        ; VECTOR string
                                06  122D
                     000001FE 00000002  1234                .LONG   ^02,^0776       ; 2 , 776 limits on input
                                82  123C                .BYTE   ^X82                    ; Indicate scan decimal
                           52 42 00'  123D                .ASCIC  "BR"    ; BR string
                                02  123D
                     00000007 00000004  1240                .LONG   4,7     ; 4 , 7 limits on input
                                81  1248                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                    1249    335 TU45:   $DS_TU45
                                    1249                .SAVE   LOCAL_BLOCK
                                    1249                .PSECT  $ABS$,ABS
                         00000032  0046                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,HP$K_TU45_LEN,       HP$K_TU45_LEN:
                                   0032                .IIF    NB,TU45$K_LEN,  TU45$K_LEN:
                              00001249                 .RESTORE
                     35 34 55 54 00'  1249                .ASCIC  "TU45"  ; TU45 name
                                04  1249
                                08  124E                .BYTE   8                       ; Maximum unit number
```

```
                        80  124F                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1250                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1251    336 TU58:   $DS_TU58
                            1251                .SAVE   LOCAL_BLOCK
                            1251                .PSECT  $ABS$,ABS
                    00000032  0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_TU58_LEN,       HP$K_TU58_LEN:
                            0032                .IIF    NB,TU58$K_LEN,  TU58$K_LEN:
                        00001251                .RESTORE
                 38 35 55 54 00'  1251          .ASCIC  "TU58"  ; TU58 name
                        04  1251
                        02  1256                .BYTE   2                 ; Maximum unit number
                        80  1257                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1258                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1259    337 TU77:   $DS_TU77
                            1259                .SAVE   LOCAL_BLOCK
                            1259                .PSECT  $ABS$,ABS
                    00000032  0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_TU77_LEN,       HP$K_TU77_LEN:
                            0032                .IIF    NB,TU77$K_LEN,  TU77$K_LEN:
                        00001259                .RESTORE
                 37 37 55 54 00'  1259          .ASCIC  "TU77"  ; TU77 name
                        04  1259
                        08  125E                .BYTE   8                 ; Maximum unit number
                        80  125F                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1260                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1261    338 TU78:   $DS_TU78                        ;                                       [01]
                            1261                .SAVE   LOCAL_BLOCK
                            1261                .PSECT  $ABS$,ABS
                    00000032  0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_TU78_LEN,       HP$K_TU78_LEN:
                            0032                .IIF    NB,TU78$K_LEN,  TU78$K_LEN:
                        00001261                .RESTORE
                 38 37 55 54 00'  1261          .ASCIC  "TU78"  ; TU78 name
                        04  1261
                        08  1266                .BYTE   8                 ; Maximum unit number
                        80  1267                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1268                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1269    339 TU80:   $DS_TU80                        ;                                       [08]
                            1269                .SAVE   LOCAL_BLOCK
                            1269                .PSECT  $ABS$,ABS
                    00000032  0046                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$L_TU80_CSR,       HP$L_TU80_CSR:
                    00000036  0032                .IIF    NL,.BLKL,       .BLKL   1
                            0036                .IIF    NB,TU80$B_BR,   TU80$B_BR:
                            0036                .IIF    NB,HP$B_TU80_BR,        HP$B_TU80_BR:
                    00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                            0037                .IIF    NB,HP$K_TU80_LEN,       HP$K_TU80_LEN:
                            0037                .IIF    NB,TU80$K_LEN,  TU80$K_LEN:
                        00001269                .RESTORE
                 30 38 55 54 00'  1269          .ASCIC  "TU80"  ; TU80 name
                        04  1269
                        08  126E                .BYTE   8                 ; Maximum unit number
                        80  126F                .BYTE   ^X80              ; Constant to identify start of descriptor
                        83  1270                .BYTE   ^X83              ; Indicate scan octal
44 41 5F 52 53 43 5F 30 38 55 54 00'  1271      .ASCIC  "TU80_CSR_ADDRESS"      ; TU80_CSR_ADDRESS string
                 53 53 45 52 44  127D
```

```
                                    10  1271
                0003FFFE 0003E000   82  1282                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                    83  128A                .BYTE   ^X83                    ; Indicate scan octal
52 4F 54 43 45 56 5F 30 38 55 54 00' 0B  128B                .ASCIC  ''TU80_VECTOR''         ; TU80_VECTOR string
                                    0B  128B
                000001FE 00000002   82  1297                .LONG   ^O2,^O776               ; 2 , 776 limits on input
                                    82  129F                .BYTE   ^X82                    ; Indicate scan decimal
        4C 45 56 45 4C 5F 52 42 00' 08  12A0                .ASCIC  ''BR_LEVEL''            ; BR_LEVEL string
                                    08  12A0
                00000007 00000004   81  12A9                .LONG   4,7     ; 4 , 7 limits on input
                                    81  12B1                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                        12B2    340 TU81:   $DS_TU81                        ;                                       [09]
                                        12B2                .SAVE   LOCAL_BLOCK
                                        12B2                .PSECT  $ABS$,ABS
                                00000032 C046               .=HP$A_DEPENDENT
                                        0032                .IIF    NB,HP$L_TU81_CSR,       HP$L_TU81_CSR:
                                        0032                .IIF    NB,TU81$L_CSR,  TU81$L_CSR:
                                00000036 0032               .IIF    NB,.BLKL,       .BLKL   1
                                        0036                .IIF    NB,HP$B_TU81_BR,        HP$B_TU81_BR:
                                        0036                .IIF    NB,TU81$B_BR,   TU81$B_BR:
                                00000037 0036               .IIF    NB,.BLKB,       .BLKB   1
                                        0037                .IIF    NB,HP$K_TU81_LEN,       HP$K_TU81_LEN:
                                        0037                .IIF    NB,TU81$K_LEN,  TU81$K_LEN:
                                    00001282                .RESTORE
                    31 38 55 54 00' 1282                .ASCIC  ''TU81''  ; TU81 name
                                    04  1282
                                    04  1287                .BYTE   4                       ; Maximum unit number
                                    80  1288                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                    83  1289                .BYTE   ^X83                    ; Indicate scan octal
                        52 53 43 00' 128A                   .ASCIC  ''CSR''   ; CSR string
                                    03  128A
                    0003FFFE 0003E000   128E                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                    83  1296                .BYTE   ^X83                    ; Indicate scan octal
                52 4F 54 43 45 56 00' 1297                  .ASCIC  ''VECTOR''              ; VECTOR string
                                    06  1297
                    000001FE 00000002   129E                .LONG   ^O2,^O776               ; 2 , 776 limits on input
                                    82  12A6                .BYTE   ^X82                    ; Indicate scan decimal
                            52 42 00' 12A7                  .ASCIC  ''BR''    ; BR string
                                    02  12A7
                    00000007 00000004   12AA                .LONG   4,7     ; 4 , 7 limits on input
                                    81  12B2                .BYTE   ^X81                    ; Indicate end of PT-descriptor
                                        12B3    341 UBE:    $DS_UBE
                                        12B3                .SAVE   LOCAL_BLOCK
                                        12B3                .PSECT  $ABS$,ABS
                                00000032 0046               .=HP$A_DEPENDENT
                                        0032                .IIF    NB,HP$L_UBE_CSR,        HP$L_UBE_CSR:
                                        0032                .IIF    NB,UBE$L_CSR,   UBE$L_CSR:
                                00000036 0032               .IIF    NB,.BLKL,       .BLKL   1
                                        0036                .IIF    NB,HP$K_UBE_LEN,        HP$K_UBE_LEN:
                                        0036                .IIF    NB,UBE$K_LEN,   UBE$K_LEN:
                                    000012E3                .RESTORE
                        45 42 55 00' 12E3                   .ASCIC  ''UBE''   ; UBE name
                                    03  12E3
                                    08  12E7                .BYTE   8                       ; Maximum unit number
                                    80  12E8                .BYTE   ^X80                    ; Constant to identify start of descriptor
                                    83  12E9                .BYTE   ^X83                    ; Indicate scan octal
                        52 53 43 00' 12EA                   .ASCIC  ''CSR''   ; CSR string
```

```
                        03  12EA
               0003FFFE 0003E000  12EE                  .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        83  12F6                  .BYTE   ^X83            ; Indicate scan octal
            52 4F 54 43 45 56 00'  12F7                  .ASCIC  "VECTOR"        ; VECTOR string
                        06  12F7
               000001FE 00000002  12FE                  .LONG   ^O2,^O776       ; 2 , 776 limits on input
                        81  1306                  .BYTE   ^X81            ; Indicate end of PT-descriptor
                            1307    342 UDA50:    $DS_UDA50
                            1307                  .SAVE   LOCAL_BLOCK
                            1307                  .PSECT  $ABS$,ABS
                    00000032  0046                  .=HP$A_DEPENDENT
                            0032                  .IIF    NB,HP$L_UDA50_UDAIP,    HP$L_UDA50_UDAIP:
                            0032                  .IIF    NB,UDA50$L_UDAIP,       UDA50$L_UDAIP:
                    00000036  0032                  .IIF    NB,.BLKL,       .BLKL   1
                            0036                  .IIF    NB,HP$B_UDA50_BR,       HP$B_UDA50_BR:
                            0036                  .IIF    NB,UDA50$B_BR,  UDA50$B_BR:
                    00000037  0036                  .IIF    NB,.BLKB,       .BLKB   1
                            0037                  .IIF    NB,HP$B_UDA50_BURST,    HP$B_UDA50_BURST:
                            0037                  .IIF    NB,UDA50$B_BURST,       UDA50$B_BURST:
                    00000038  0037                  .IIF    NB,.BLKB,       .BLKB   1
                            0038                  .IIF    NB,HP$K_UDA50_LEN,      HP$K_UDA50_LEN:
                            0038                  .IIF    NB,UDA50$K_LEN, UDA50$K_LEN:
                        00001307                  .RESTORE
               30 35 41 44 55 00'  1307                  .ASCIC  "UDA50" ; UDA50 name
                        05  1307
                        00  130D                  .BYTE   0               ; Maximum unit number
                        80  130E                  .BYTE   ^X80            ; Constant to identify start of descriptor
                        83  130F                  .BYTE   ^X83            ; Indicate scan octal
               50 49 41 44 55 00'  1310                  .ASCIC  "UDAIP" ; UDAIP string
                        05  1310
               0003FFFC 0003E000  1316                  .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                        83  131E                  .BYTE   ^X83            ; Indicate scan octal
            72 6F 74 63 65 56 00'  131F                  .ASCIC  "Vector"        ; Vector string
                        06  131F
               000001FC 00000004  1326                  .LONG   ^O4,^O774       ; 4 , 774 limits on input
                        82  132E                  .BYTE   ^X82            ; Indicate scan decimal
                    52 42 00'  132F                  .ASCIC  "BR"    ; BR string
                        02  132F
                00000007 00000004  1332                  .LONG   4,7     ; 4 , 7 limits on input
                        82  133A                  .BYTE   ^X82            ; Indicate scan decimal
         65 74 61 52 5F 74 73 72 75 42 00'  133B                  .ASCIC  "Burst_Rate"    ; Burst_Rate string
                        0A  133B
               0000003F 00000000  1346                  .LONG   0,63    ; 0 , 63 limits on input
                        81  134E                  .BYTE   ^X81            ; Indicate end of PT-descriptor
                            134F    343 UNA11:    $DS_UNA11                       ;                                       [06]
                            134F                  .SAVE   LOCAL_BLOCK
                            134F                  .PSECT  $ABS$,ABS
                    00000032  0046                  .=HP$A_DEPENDENT
                            0032                  .IIF    NB,UNA11$B_BR,  UNA11$B_BR:
                    00000033  0032                  .IIF    NB,.BLKB,       .BLKB   1
                            0033                  .IIF    NB,UNA11$L_CSR, UNA11$L_CSR:
                    00000037  0033                  .IIF    NB,.BLKL,       .BLKL   1
                            0037                  .IIF    NB,UNA11$K_LEN, UNA11$K_LEN:
                        0000134F                  .RESTORE
               31 31 41 4E 55 00'  134F                  .ASCIC  "UNA11" ; UNA11 name
                        05  134F
                        01  1355                  .BYTE   1               ; Maximum unit number
```

```
                        80  1356                .BYTE   ^X80            ; Constant to identify start of descriptor
                        83  1357                .BYTE   ^X83            ; Indicate scan octal
                 52 53 43 00' 1358                .ASCIC  "CSR"   ; CSR string
                        03  1358
              0003FFFE 0003E000  135C            .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                        83  1364                .BYTE   ^X83            ; Indicate scan octal
           52 4F 54 43 45 56 00' 1365           .ASCIC  "VECTOR"        ; VECTOR string
                        06  1365
              000001FE 00000002  136C           .LONG   ^02,^0776       ; 2 , 776 limits on input
                        82  1374                .BYTE   ^X82            ; Indicate scan decimal
                    52 42 00' 1375              .ASCIC  "BR"    ; BR string
                        02  1375
              00000007 00000004  1378           .LONG   4,7     ; 4 , 7 limits on input
                        81  1380                .BYTE   ^X81            ; Indicate end of PT-descriptor
                            1381    344 VS100:  $DS_VS100                       ;                                   [10]
                            1381                .SAVE   LOCAL_BLOCK
                            1381                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,HP$L_VS100_CSR,      HP$L_VS100_CSR:
                              0032              .IIF    NB,VS100$L_CSR, VS100$L_CSR:
                    00000036  0032              .IIF    NB,.BLKL,       .BLKL   1
                              0036              .IIF    NB,HP$B_VS100_BR,       HP$B_VS100_BR:
                              0036              .IIF    NB,VS100$B_BR,  VS100$B_BR:
                    00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,HP$K_VS100_LEN,      HP$K_VS100_LEN:
                              0037              .IIF    NB,VS100$K_LEN, VS100$K_LEN:
                          00001381              .RESTORE
              30 30 31 53 56 00' 1381           .ASCIC  "VS100" ; VS100 name
                        05  1381
                        01  1387                .BYTE   1               ; Maximum unit number
                        80  1388                .BYTE   ^X80            ; Constant to identify start of descriptor
                        83  1389                .BYTE   ^X83            ; Indicate scan octal
                 52 53 43 00' 138A                .ASCIC  "CSR"   ; CSR string
                        03  138A
              0003FFFE 0003E000  138E           .LONG   ^0760000,^0777776       ; 760000 , 777776 limits on input
                        83  1396                .BYTE   ^X83            ; Indicate scan octal
           52 4F 54 43 45 56 00' 1397           .ASCIC  "VECTOR"        ; VECTOR string
                        06  1397
              000001FE 00000002  139E           .LONG   ^02,^0776       ; 2 , 776 limits on input
                        82  13A6                .BYTE   ^X82            ; Indicate scan decimal
                    52 42 00' 13A7              .ASCIC  "BR"    ; BR string
                        02  13A7
              00000007 00000004  13AA           .LONG   4,7     ; 4 , 7 limits on input
                        81  13B2                .BYTE   ^X81            ; Indicate end of PT-descriptor
                            13B3    345 VS125:  $DS_VS125                       ;                                   [17]
                            13B3                .SAVE   LOCAL_BLOCK
                            13B3                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                              0032              .IIF    NB,HP$L_VS125_CSR,      HP$L_VS125_CSR:
                              0032              .IIF    NB,VS125$L_CSR, VS125$L_CSR:
                    00000036  0032              .IIF    NB,.BLKL,       .BLKL   1
                              0036              .IIF    NB,HP$B_VS125_BR,       HP$B_VS125_BR:
                              0036              .IIF    NB,VS125$B_BR,  VS125$B_BR:
                    00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,HP$K_VS125_LEN,      HP$K_VS125_LEN:
                              0037              .IIF    NB,VS125$K_LEN, VS125$K_LEN:
                          000013B3              .RESTORE
```

```
                35 32 31 53 56 00' 13B3              .ASCIC  "VS125" ; VS125 name
                                05 13B3
                                01 13B9              .BYTE   1               ; Maximum unit number
                                80 13BA              .BYTE   ^X80            ; Constant to identify start of descriptor
                                83 13BB              .BYTE   ^X83            ; Indicate scan octal
                      52 53 43 00' 13BC              .ASCIC  "CSR"   ; CSR string
                                03 13BC
                0003FFFE 0003E000  13C0              .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                83 13C8              .BYTE   ^X83            ; Indicate scan octal
             52 4F 54 43 45 56 00' 13C9              .ASCIC  "VECTOR"        ; VECTOR string
                                06 13C9
                000001FE 00000002  13D0              .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                82 13D8              .BYTE   ^X82            ; Indicate scan decimal
                          52 42 00' 13D9             .ASCIC  "BR"    ; BR string
                                02 13D9
                00000007 00000004  13DC              .LONG   4,7     ; 4 , 7 limits on input
                                81 13E4              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                   13E5  346 VS300:  $DS_VS300                       ;                                [12]
                                   13E5              .SAVE   LOCAL_BLOCK
                                   13E5              .PSECT  $ABS$,ABS
                          00000032 0046              .=HP$A_DEPENDENT
                                   0032              .IIF    NB,HP$L_VS300_CSR,      HP$L_VS300_CSR:
                                   0032              .IIF    NB,VS300$L_CSR, VS300$L_CSR:
                          00000036 0032              .IIF    NB,.BLKL,       .BLKL   1
                                   0036              .IIF    NB,HP$B_VS300_BR,       HP$B_VS300_BR:
                                   0036              .IIF    NB,VS300$B_BR,  VS300$B_BR:
                          00000037 0036              .IIF    NB,.BLKB,       .BLKB   1
                                   0037              .IIF    NB,HP$K_VS300_LEN,      HP$K_VS300_LEN:
                                   0037              .IIF    NB,VS300$K_LEN, VS300$K_LEN:
                                0000 13E5            .RESTORE
                30 30 33 53 56 00' 13E5              .ASCIC  "VS300" ; VS300 name
                                05 13E5
                                01 13EB              .BYTE   1               ; Maximum unit number
                                80 13EC              .BYTE   ^X80            ; Constant to identify start of descriptor
                                83 13ED              .BYTE   ^X83            ; Indicate scan octal
                      52 53 43 00' 13EE              .ASCIC  "CSR"   ; CSR string
                                03 13EE
                0003FFFE 0003E000  13F2              .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                83 13FA              .BYTE   ^X83            ; Indicate scan octal
             52 4F 54 43 45 56 00' 13FB              .ASCIC  "VECTOR"        ; VECTOR string
                                06 13FB
                000001FE 00000002  1402              .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                82 140A              .BYTE   ^X82            ; Indicate scan decimal
                          52 42 00' 140B             .ASCIC  "BR"    ; BR string
                                02 140B
                00000007 00000004  140E              .LONG   4,7     ; 4 , 7 limits on input
                                81 1416              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                   1417  347 VT50:   $DS_VT50
                                   1417              .SAVE   LOCAL_BLOCK
                                   1417              .PSECT  $ABS$,ABS
                          00000032 0046              .=HP$A_DEPENDENT
                                   0032              .IIF    NB,HP$K_VT50_LEN,       HP$K_VT50_LEN:
                                   0032              .IIF    NB,VT50$K_LEN,  VT50$K_LEN:
                                0000 1417            .RESTORE
                   30 35 54 56 00' 1417              .ASCIC  "VT50"  ; VT50 name
                                04 1417
                                08 141C              .BYTE   8               ; Maximum unit number
```

```
                        80  141D                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  141E                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            141F    348 VT52:   $DS_VT52
                            141F                .SAVE   LOCAL_BLOCK
                            141F                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_VT52_LEN,       HP$K_VT52_LEN:
                            0032                .IIF    NB,VT52$K_LEN,  VT52$K_LEN:
                        0000141F                .RESTORE
                32 35 54 56 00' 141F            .ASCIC  "VT52"  ; VT52 name
                        04  141F
                        08  1424                .BYTE   8                 ; Maximum unit number
                        80  1425                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1426                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1427    349 VT55:   $DS_VT55
                            1427                .SAVE   LOCAL_BLOCK
                            1427                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_VT55_LEN,       HP$K_VT55_LEN:
                            0032                .IIF    NB,VT55$K_LEN,  VT55$K_LEN:
                        00001427                .RESTORE
                35 35 54 56 00' 1427            .ASCIC  "VT55"  ; VT55 name
                        04  1427
                        08  142C                .BYTE   8                 ; Maximum unit number
                        80  142D                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  142E                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            142F    350 VT100:  $DS_VT100
                            142F                .SAVE   LOCAL_BLOCK
                            142F                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_VT100_LEN,      HP$K_VT100_LEN:
                            0032                .IIF    NB,VT100$K_LEN, VT100$K_LEN:
                        0000142F                .RESTORE
            30 30 31 54 56 00' 142F             .ASCIC  "VT100" ; VT100 name
                        05  142F
                        08  1435                .BYTE   8                 ; Maximum unit number
                        80  1436                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1437                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1438    351 VT220:  $DS_VT220                 ;                                          [14]
                            1438                .SAVE   LOCAL_BLOCK
                            1438                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_VT220_LEN,      HP$K_VT220_LEN:
                            0032                .IIF    NB,VT220$K_LEN, VT220$K_LEN:
                        00001438                .RESTORE
            30 32 32 54 56 00' 1438             .ASCIC  "VT220" ; VT220 name
                        05  1438
                        08  143E                .BYTE   8                 ; Maximum unit number
                        80  143F                .BYTE   ^X80              ; Constant to identify start of descriptor
                        81  1440                .BYTE   ^X81              ; Indicate end of PT-descriptor
                            1441    352 VT240:  $DS_VT240                 ;                                          [14]
                            1441                .SAVE   LOCAL_BLOCK
                            1441                .PSECT  $ABS$,ABS
                    00000032  0046              .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_VT240_LEN,      HP$K_VT240_LEN:
                            0032                .IIF    NB,VT240$K_LEN, VT240$K_LEN:
                        00001441                .RESTORE
```

```
                30 34 32 54 56 00' 1441              .ASCIC  "VT240" ; VT240 name
                                05 1441
                                08 1447              .BYTE   8               ; Maximum unit number
                                80 1448              .BYTE   ^X80            ; Constant to identify start of descriptor
                                81 1449              .BYTE   ^X81            ; Indicate end of PT-descriptor
                                   144A   353        .END
```

```
$$N                     = 00000076     D
AA11K                     000000EE R   D   01
AA11K$B_BR                00000032     D
AA11K$K_LEN               00000033     D
AD11K                     00000128 R   D   01
AD11K$B_BR                00000032     D
AD11K$K_LEN               00000033     D
AL_DEVTYP                 00000000 R   D   01
CI750                     000001B0 R   D   01
CI780                     000001E1 R   D   01
CI_NODE                   00000162 R   D   01
CONSOLE                   00000210 R   D   01
CONSOLE$K_LEN             00000032     D
CR11                      0000021B R   D   01
CR11$B_BR                 00000036     D
CR11$K_LEN                00000037     D
CR11$L_CSR                00000032     D
DEQNA                     0000024C R   D   01
DEQNA$K_LEN               00000036     D
DEQNA$L_CSR               00000032     D
DISK                      00000262 R   D   01
DISK$K_LEN                00000032     D
DL11                      0000026A R   D   01
DL11$B_BR                 00000036     D
DL11$K_LEN                00000037     D
DL11$L_CSR                00000032     D
DLVJ1                     0000029B R   D   01
DLVJ1$B_FLAGS             00000036     D
DLVJ1$K_LEN               00000037     D
DLVJ1$L_CSR               00000032     D
DMC11                     00000334 R   D   01
DMC11$B_BR                00000036     D
DMC11$K_LEN               00000037     D
DMC11$L_CSR               00000032     D
DMF32                     00000366 R   D   01
DMF32$B_BR                00000034     D
DMF32$B_COMMON            00000032     D
DMF32$B_FLAGS             00000033     D
DMF32$K_LEN               00000039     D
DMF32$L_CSR               00000035     D
DMF32A                    00000398 R   D   01
DMF32A$B_ACTIV            00000033     D
DMF32A$B_BR               00000032     D
DMF32A$B_JUMP             00000037     D
DMF32A$B_SPEED            00000034     D
DMF32A$K_LEN              00000038     D
DMF32A$W_FLAGS            00000035     D
DMF32P                    00000497 R   D   01
DMF32P$B_BR               00000034     D
DMF32P$B_COMMON           00000032     D
DMF32P$B_FLAGS            00000033     D
DMF32P$K_LEN              00000039     D
DMF32P$L_CSR              00000035     D
DMF32S                    000004E6 R   D   01
DMF32S$B_BR               00000034     D
DMF32S$B_COMMON           00000032     D
DMF32S$B_FLAGS            00000033     D
DMF32S$K_LEN              00000039     D
DMF32S$L_CSR              00000035     D
DMP11                     00000539 R   D   01
DMP11$B_BR                00000034     D
DMP11$B_COMMON            00000032     D
DMP11$B_FLAGS             00000033     D
DMP11$K_LEN               00000039     D
DMP11$L_CSR               00000035     D
DMR11                     000005C7 R   D   01
DMR11$B_BR                00000036     D
DMR11$K_LEN               00000037     D
DMR11$L_CSR               00000032     D
DMZ32                     000005F9 R   D   01
DR11B                     00000725 R   D   01
DR11B$B_BR                00000036     D
DR11B$K_LEN               00000037     D
DR11B$L_CSR               00000032     D
DR11K                     00000757 R   D   01
DR11K$B_BR                00000032     D
DR11K$K_LEN               00000033     D
DR11W                     00000791 R   D   01
DR11W$B_BR                00000036     D
DR11W$K_LEN               00000037     D
DR11W$L_CSR               00000032     D
DR750                     000007C3 R   D   01
DR750$B_BR                00000033     D
DR750$B_CONFIG            00000035     D
DR750$B_SELF              00000034     D
DR750$B_SLOT              00000032     D
DR750$B_UUT               00000036     D
DR750$K_LEN               00000037     D
DR780                     00000824 R   D   01
DR780$B_BR                00000033     D
DR780$B_CONFIG            00000035     D
DR780$B_SELF              00000034     D
DR780$B_TR                00000032     D
DR780$B_UUT               00000036     D
DR780$K_LEN               00000037     D
DS$GA_PTDESC              00000000 RG  D   01
DUP11                     00000883 R   D   01
DUP11$B_BR                00000036     D
DUP11$K_LEN               00000037     D
DUP11$L_CSR               00000032     D
DV11                      000008B5 R   D   01
DV11$B_BR                 00000036     D
DV11$K_LEN                00000037     D
DV11$L_CSR                00000032     D
DW730                     000008E6 R   D   01
DW730$K_LEN               00000032     D
DW750                     000008EF R   D   01
DW750$K_LEN               00000032     D
DW780                     000008F8 R   D   01
DW780$B_BR                00000033     D
DW780$B_TR                00000032     D
DW780$K_LEN               00000034     D
DZ11                      00000919 R   D   01
DZ11$B_BR                 00000036     D
```

```
DZ11$B_MTYPE                    00000037    D
DZ11$K_LEN                      00000038    D
DZ11$L_CSR                      00000032    D
DZ32                            00000961 R  D  01
DZ32$B_ACTIV                    00000033    D
DZ32$B_BRLVL                    00000032    D
DZ32$B_SPEED                    00000034    D
DZ32$W_FLAGS                    00000035    D
DZ32$_LEN                       00000037    D
DZV11                           00000A1D R  D  01
DZV11$K_LEN                     00000036    D
DZV11$L_CSR                     00000032    D
HP$A_DEPENDENT                  00000032    D
HP$A_DEVICE                     00000018    D
HP$A_DVA                        0000C01C    D
HP$A_LINK                       00000020    D
HP$B_DMF32A_ACTIV               00000033    D
HP$B_DMF32A_BR                  00000032    D
HP$B_DMF32A_JUMP                00000037    D
HP$B_DMF32A_SPEED               00000034    D
HP$B_DMF32P_BR                  00000034    D
HP$B_DMF32P_COMMON              00000032    D
HP$B_DMF32P_FLAGS               00000033    D
HP$B_DMF32S_BR                  00000034    D
HP$B_DMF32S_COMMON              00000032    D
HP$B_DMF32S_FLAGS               00000033    D
HP$B_DMF32_BR                   00000034    D
HP$B_DMF32_COMMON               00000032    D
HP$B_DMF32_FLAGS                00000033    D
HP$B_DMZ32_BR                   00000032    D
HP$B_DMZ32_MODEM                00000039    D
HP$B_DMZ32_OCTET0               00000033    D
HP$B_DMZ32_OCTET1               00000034    D
HP$B_DMZ32_OCTET2               00000035    D
HP$B_DMZ32_SPEED                00000036    D
HP$B_DRIVE                      0000000B    D
HP$B_DZ32_ACTIV                 00000033    D
HP$B_DZ32_BRLVL                 00000032    D
HP$B_DZ32_SPEED                 00000034    D
HP$B_FLAGS                      0000000A    D
HP$B_IEU11A_BR                  00000032    D
HP$B_LESI_BR                    00000036    D
HP$B_RB730_BR                   00000036    D
HP$B_RH750_BR                   00000032    D
HP$B_RH780_BR                   00000033    D
HP$B_RH780_TR                   00000032    D
HP$B_RK611_BR                   00000036    D
HP$B_RL11_BR                    00000036    D
HP$B_RX21T_BR                   00000036    D
HP$B_TM03_DRIVE                 00000032    D
HP$B_TM78_DRIVE                 00000032    D
HP$B_TS04_BR                    00000036    D
HP$B_TS11_BR                    00000036    D
HP$B_TU80_BR                    00000036    D
HP$B_TU81_BR                    00000036    D
HP$B_UDA50_BR                   00000036    D
HP$B_UDA50_BURST                00000037    D
HP$B_VS100_BR                   00000036    D
HP$B_VS125_BR                   00000036    D
HP$B_VS300_BR                   00000036    D
HP$K_DISK_LEN                   00000032    D
HP$K_DMF32A_LEN                 00000038    D
HP$K_DMF32P_LEN                 00000039    D
HP$K_DMF32S_LEN                 00000039    D
HP$K_DMF32_LEN                  00000039    D
HP$K_DMZ32_LEN                  0000003A    D
HP$K_IEU11A_LEN                 00000033    D
HP$K_LA100_LEN                  00000032    D
HP$K_LA12_LEN                   00000032    D
HP$K_LEN                        00000037    D
HP$K_LESI_LEN                   00000037    D
HP$K_RA60_LEN                   00000032    D
HP$K_RA80_LEN                   00000032    D
HP$K_RA81_LEN                   00000032    D
HP$K_RB730_LEN                  00000037    D
HP$K_RC25_LEN                   00000032    D
HP$K_RCF25_LEN                  00000032    D
HP$K_RH750_LEN                  00000033    D
HP$K_RH780_LEN                  00000034    D
HP$K_RK06_LEN                   00000032    D
HP$K_RK07_LEN                   00000032    D
HP$K_RK611_LEN                  00000037    D
HP$K_RL01_LEN                   00000032    D
HP$K_RL02_LEN                   00000032    D
HP$K_RL11_LEN                   00000037    D
HP$K_RM03_LEN                   00000032    D
HP$K_RM05_LEN                   00000032    D
HP$K_RM80_LEN                   00000032    D
HP$K_RP04_LEN                   00000032    D
HP$K_RP05_LEN                   00000032    D
HP$K_RP06_LEN                   00000032    D
HP$K_RP07_LEN                   00000032    D
HP$K_RX02_LEN                   00000032    D
HP$K_RX21T_LEN                  00000037    D
HP$K_SBIA_LEN                   00000032    D
HP$K_TE16_LEN                   00000032    D
HP$K_TM03_LEN                   00000033    D
HP$K_TM78_LEN                   00000033    D
HP$K_TS04_LEN                   00000037    D
HP$K_TS05_LEN                   00000033    D
HP$K_TS11_LEN                   00000037    D
HP$K_TU45_LEN                   00000032    D
HP$K_TU58_LEN                   00000032    D
HP$K_TU77_LEN                   00000032    D
HP$K_TU78_LEN                   00000032    D
HP$K_TU80_LEN                   00000037    D
HP$K_TU81_LEN                   00000037    D
HP$K_UBE_LEN                    00000036    D
HP$K_UDA50_LEN                  00000038    D
HP$K_VS100_LEN                  00000037    D
HP$K_VS125_LEN                  00000037    D
HP$K_VS300_LEN                  00000037    D
HP$K_VT100_LEN                  00000032    D
HP$K_VT220_LEN                  00000032    D
```

```
HP$K_VT240_LEN                  00000032    D
HP$K_VT50_LEN                   00000032    D
HP$K_VT52_LEN                   00000032    D
HP$K_VT55_LEN                   00000032    D
HP$L_DMF32P_CSR                 00000035    D
HP$L_DMF32S_CSR                 00000035    D
HP$L_DMF32_CSR                  00000035    D
HP$L_LESI_IP                    00000032    D
HP$L_RB730_CSR                  00000032    D
HP$L_RK611_CSR                  00000032    D
HP$L_RL11_CSR                   00000032    D
HP$L_RX21T_CSR                  00000032    D
HP$L_TS04_CSR                   00000032    D
HP$L_TS11_CSR                   00000032    D
HP$L_TU80_CSR                   00000032    D
HP$L_TU81_CSR                   00000032    D
HP$L_UBE_CSR                    00000032    D
HP$L_UDA50_UDAIP                00000032    D
HP$L_VS100_CSR                  00000032    D
HP$L_VS125_CSR                  00000032    D
HP$L_VS300_CSR                  00000032    D
HP$M_ALLOC                    = 00000001    D
HP$Q_DEVICE                     00000000    D
HP$T_DEVICE                     0000000C    D
HP$T_TYPE                       00000026    D
HP$W_DMF32A_FLAGS               00000035    D
HP$W_DMZ32_LOOP                 00000037    D
HP$W_DZ32_FLAGS                 00000035    D
HP$W_SIZE                       00000008    D
HP$W_VECTOR                     00000024    D
IEU11A                          00000A43 R  D  01
IEU11A$B_BR                     00000032    D
IEU11A$K_LEN                    00000033    D
KA$B_ACC_TYPE                   00000042    D
KA$B_RESERVED                   00000043    D
KA$B_SID_TYPE                   0000003D    D
KA$K_LEN                        00000046    D
KA$L_FLAGS                      00000032    D
KA$L_SID                        0000003A    D
KA$M_CHAR                     = 00000100    D
KA$M_CRC                      = 00000010    D
KA$M_D_FLOAT                  = 00000002    D
KA$M_EDIT                     = 00000080    D
KA$M_F_FLOAT                  = 00000001    D
KA$M_G_FLOAT                  = 00000004    D
KA$M_H_FLOAT                  = 00000008    D
KA$M_PACKED                   = 00000020    D
KA$M_PACK_MUL                 = 00000040    D
KA$M_TODR                     = 00010000    D
KA$W_LATENCY                    0000003E    D
KA$W_MEM_SIZE                   00000044    D
KA$W_PROGRESS                   00000040    D
KA$W_RESERVED                   0000C038    D
KA$W_WCS                        00000036    D
KA730                           00000A77 R  D  01
KA750                           00000B1C R  D  01
KA780                           00000BB7 R  D  01
KA785                           00000CB5 R  D  01
KAXXX                           00000C36 R  D  01
KMC11                           00000D34 R  D  01
KMC11$B_BR                      00000036    D
KMC11$K_LEN                     00000037    D
KMC11$L_CSR                     00000032    D
KW11K                           00000D66 R  D  01
KW11K$B_BR                      00000032    D
KW11K$K_LEN                     00000033    D
LA100                           00000DA8 R  D  01
LA100$K_LEN                     00000032    D
LA12                            00000DA0 R  D  01
LA12$K_LEN                      00000032    D
LA120                           00000DB1 R  D  01
LA120$K_LEN                     00000032    D
LA180                           00000DBA R  D  01
LA180$K_LEN                     00000032    D
LA34                            00000DC3 R  D  01
LA34$K_LEN                      00000032    D
LA36                            00000DCB R  D  01
LA36$K_LEN                      00000032    D
LA38                            00000DD3 R  D  01
LA38$K_LEN                      0000C032    D
LESI                            00000DDB R  D  01
LESI$B_BR                       00000036    D
LESI$K_LEN                      00000037    D
LESI$L_IP                       00000032    D
LN01                            00000E0B R  D  01
LN01$K_LEN                      00000032    D
LP04                            00000E13 R  D  01
LP04$K_LEN                      00000032    D
LP05                            00000E1B R  D  01
LP05$K_LEN                      00000032    D
LP06                            00000E23 R  D  01
LP06$K_LEN                      00000032    D
LP07                            00000E2B R  D  01
LP07$K_LEN                      00000032    D
LP11                            00000E33 R  D  01
LP11$B_BR                       00000036    D
LP11$K_LEN                      00000037    D
LP11$L_CSR                      00000032    D
LP14                            00000E64 R  D  01
LP14$K_LEN                      00000032    D
LP25                            00000E6C R  D  01
LP25$K_LEN                      00000032    D
LP26                            00000E74 R  D  01
LP26$K_LEN                      00000032    D
LP27                            00000E7C R  D  01
LP27$K_LEN                      00000032    D
LPA11K                          00000E84 R  D  01
LPA11K$B_BR                     00000036    D
LPA11K$K_LEN                    00000037    D
LPA11K$L_CSR                    00000032    D
MA780                           00000EB7 R  D  01
MA780$B_BR                      00000033    D
MA780$B_MPM                     00000034    D
MA780$B_PORT                    00000035    D
```

```
MA780$B_TR                      00000032     D
MA780$K_LEN                     00000036     D
MBE                             00000EF3 R   D  01
MBE$K_LEN                       00000032     D
ML11                            00000EFA R   D  01
ML11$B_ARRAY                    00000032     D
ML11$B_CHIP                     00000033     D
ML11$K_LEN                      00000034     D
MS750                           00000F36 R   D  01
MS750$K_LEN                     00000032     D
MS780                           00000F3F R   D  01
MS780$B_ARRAYS                  00000033     D
MS780$B_TR                      00000032     D
MS780$K_LEN                     00000034     D
PCL11                           00000F6E R   D  01
PCL11$B_BR                      00000036     D
PCL11$K_LEN                     00000037     D
PCL11$L_CSR                     00000032     D
PX780                           00000FA0 R   D  01
PX780$B_BR                      00000033     D
PX780$B_NODE                    00000034     D
PX780$B_TR                      00000032     D
PX780$K_LEN                     00000035     D
R80                             00000FCF R   D  01
R80$K_LEN                       00000032     D
RA60                            00000FD6 R   D  01
RA60$K_LEN                      00000032     D
RA80                            00000FDE R   D  01
RA80$K_LEN                      00000032     D
RA81                            00000FE6 R   D  01
RA81$K_LEN                      00000032     D
RB730                           00000FEE R   D  01
RB730$B_BR                      00000036     D
RB730$K_LEN                     00000037     D
RB730$L_CSR                     00000032     D
RC25                            00001000 R   D  01
RC25$K_LEN                      00000032     D
RCF25                           00000FF7 R   D  01
RCF25$K_LEN                     00000032     D
RD51                            00001008 R   D  01
RD51$K_LEN                      00000032     D
RD52                            00001010 R   D  01
RD52$K_LEN                      00000032     D
RH750                           00001018 R   D  01
RH750$B_BR                      00000032     D
RH750$K_LEN                     00000033     D
RH780                           0000102D R   D  01
RH780$B_BR                      00000033     D
RH780$B_TR                      00000032     D
RH780$K_LEN                     00000034     D
RK06                            0000108F R   D  01
RK06$K_LEN                      00000032     D
RK07                            00001097 R   D  01
RK07$K_LEN                      00000032     D
RK611                           0000109F R   D  01
RK611$B_BR                      00000036     D
RK611$K_LEN                     00000037     D
RK611$L_CSR                     00000032     D
RL01                            0000104E R   D  01
RL01$K_LEN                      00000032     D
RL02                            00001056 R   D  01
RL02$K_LEN                      00000032     D
RL11                            0000105E R   D  01
RL11$B_BR                       00000036     D
RL11$K_LEN                      00000037     D
RL11$L_CSR                      00000032     D
RM03                            000010D1 R   D  01
RM03$K_LEN                      00000032     D
RM05                            000010D9 R   D  01
RM05$K_LEN                      00000032     D
RM80                            000010E1 R   D  01
RM80$K_LEN                      00000032     D
RP04                            000010E9 R   D  01
RP04$K_LEN                      00000032     D
RP05                            000010F1 R   D  01
RP05$K_LEN                      00000032     D
RP06                            000010F9 R   D  01
RP06$K_LEN                      00000032     D
RP07                            00001101 R   D  01
RP07$K_LEN                      00000032     D
RQDX1                           00001109 R   D  01
RQDX1$K_LEN                     00000036     D
RQDX1$L_IP                      00000032     D
RX02                            0000111E R   D  01
RX02$K_LEN                      00000032     D
RX211                           00001126 R   D  01
RX211$B_BR                      00000036     D
RX211$K_LEN                     00000037     D
RX211$L_CSR                     00000032     D
RX50                            00001158 R   D  01
RX50$K_LEN                      00000032     D
SBIA                            00001160 R   D  01
SBIA$K_LEN                      00000032     D
SIZ...                        = 00000001     D
TE16                            00001168 R   D  01
TE16$K_LEN                      00000032     D
TM03                            00001170 R   D  01
TM03$B_DRIVE                    00000032     D
TM03$K_LEN                      00000033     D
TM78                            00001187 R   D  01
TM78$B_DRIVE                    00000032     D
TM78$K_LEN                      00000033     D
TS04                            0000119E R   D  01
TS04$B_BR                       00000036     D
TS04$K_LEN                      00000037     D
TS04$L_CSR                      00000032     D
TS05                            000011CF R   D  01
TS05$B_BR                       00000032     D
TS05$K_LEN                      00000033     D
TS11                            00001218 R   D  01
TS11$B_BR                       00000036     D
TS11$K_LEN                      00000037     D
TS11$L_CSR                      00000032     D
TU45                            00001249 R   D  01
```

```
TU45$K_LEN                      00000032    D
TU58                            00001251 R  D  01
TU58$K_LEN                      00000032    D
TU77                            00001259 R  D  01
TU77$K_LEN                      00000032    D
TU78                            00001261 R  D  01
TU78$K_LEN                      00000032    D
TU80                            00001269 R  D  01
TU80$B_BR                       00000036    D
TU80$K_LEN                      00000037    D
TU81                            000012B2 R  D  01
TU81$B_BR                       00000036    D
TU81$K_LEN                      00000037    D
TU81$L_CSR                      00000032    D
UBE                             000012E3 R  D  01
UBE$K_LEN                       00000036    D
UBE$L_CSR                       00000032    D
UDA50                           00001307 R  D  01
UDA50$B_BR                      00000036    D
UDA50$B_BURST                   00000037    D
UDA50$K_LEN                     00000038    D
UDA50$L_UDAIP                   00000032    D
UNA11                           0000134F R  D  01
UNA11$B_BR                      00000032    D
UNA11$K_LEN                     00000037    D
UNA11$L_CSR                     00000033    D
VS100                           00001381 R  D  01
VS100$B_BR                      00000036    D
VS100$K_LEN                     00000037    D
VS100$L_CSR                     00000032    D
VS125                           000013B3 R  D  01
VS125$B_BR                      00000036    D
VS125$K_LEN                     00000037    D
VS125$L_CSR                     00000032    D
VS300                           000013E5 R  D  01
VS300$B_BR                      00000036    D
VS300$K_LEN                     00000037    D
VS300$L_CSR                     00000032    D
VT100                           0000142F R  D  01
VT100$K_LEN                     00000032    D
VT220                           00001438 R  D  01
VT220$K_LEN                     00000032    D
VT240                           00001441 R  D  01
VT240$K_LEN                     00000032    D
VT50                            00001417 R  D  01
VT50$K_LEN                      00000032    D
VT52                            0000141F R  D  01
VT52$K_LEN                      00000032    D
VT55                            00001427 R  D  01
VT55$K_LEN                      00000032    D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	000000^0 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	0000144A (5194.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000046 (70.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$$$$$	=00000001	352 (4)	
$$N	=00000076	228 (3)	228 (3)
AA11K	000000EE-R	232 (4)	228 (3)
AD11K	00000128-R	233 (4)	228 (3)
AL_DEVTYP	00000000-R	228 (3)	228 (3)
CI750	000001B0-R	235 (4)	228 (3)
CI780	000001E1-R	236 (4)	228 (3)
CI_NODE	00000162-R	234 (4)	228 (3)
CONSOLE	00000210-R	237 (4)	228 (3)
CR11	0000021B-R	238 (4)	228 (3)
DEQNA	0000024C-R	239 (4)	228 (3)
DISK	00000262-R	240 (4)	228 (3)
DL11	0000026A-R	241 (4)	228 (3)
DLVJ1	0000029B-R	242 (4)	228 (3)
DMC11	00000334-R	243 (4)	228 (3)
DMF32	00000366-R	244 (4)	228 (3)
DMF32A	00000398-R	245 (4)	228 (3)
DMF32P	00000497-R	246 (4)	228 (3)
DMF32S	000004E6-R	247 (4)	228 (3)
DMP11	00000539-R	248 (4)	228 (3)
DMR11	000005C7-R	249 (4)	228 (3)
DMZ32	000005F9-R	250 (4)	228 (3)
DR11B	00000725-R	251 (4)	228 (3)
DR11K	00000757-R	252 (4)	228 (3)
DR11W	00000791-R	253 (4)	228 (3)
DR750	000007C3-R	254 (4)	228 (3)
DR780	00000824-R	255 (4)	228 (3)
DS$GA_PTDESC	00000000-R	205 (3)	
DUP11	00000883-R	256 (4)	228 (3)
DV11	000008B5-R	257 (4)	
DW730	000008E6-R	258 (4)	228 (3)
DW750	000008EF-R	259 (4)	228 (3)
DW780	000008F8-R	260 (4)	228 (3)
DZ11	00000919-R	261 (4)	228 (3)
DZ32	00000961-R	262 (4)	228 (3)
DZV11	00000A1D-R	263 (4)	228 (3)
HP$M_ALLOC	=00000001		238 (4) 273 (4) 274 (4) 275 (4)
			276 (4) 277 (4) 278 (4) 279 (4)
			281 (4) 282 (4) 283 (4) 284 (4)
			285 (4) 287 (4) 288 (4) 289 (4)
			290 (4) 291 (4) 294 (4) 299 (4)
			300 (4) 301 (4) 302 (4) 304 (4)
			306 (4) 307 (4) 308 (4) 311 (4)
			312 (4) 314 (4) 315 (4) 317 (4)
			318 (4) 319 (4) 320 (4) 321 (4)
			322 (4) 323 (4) 325 (4) 327 (4)
			329 (4) 332 (4) 333 (4) 334 (4)
			335 (4) 336 (4) 337 (4) 338 (4)
			339 (4) 340 (4) 344 (4) 345 (4)
			346 (4) 347 (4) 348 (4) 349 (4)

```
                                                  350  (4)     351  (4)     352  (4)
IEU11A                          00000A43-R  264   (4)  228  (3)
KA$M_CHAR                       =00000100           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_CRC                        =00000010           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_D_FLOAT                    =00000002           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_EDIT                       =00000080           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_F_FLOAT                    =00000001           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_G_FLOAT                    =00000004           266  (4)
KA$M_H_FLOAT                    =00000008           266  (4)
KA$M_PACKED                     =00000C20           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_PACK_MUL                   =00000040           266  (4)     267  (4)     268  (4)     269  (4)
                                                    270  (4)
KA$M_TODR                       =00010000           268  (4)     269  (4)     270  (4)
KA730                            00000A77-R  266   (4)  228  (3)
KA750                            00000B1C-R  267   (4)  228  (3)
KA780                            00000BB7-R  268   (4)  228  (3)
KA785                            00000CB5-R  270   (4)  228  (3)
KAXXX                            00000C36-R  269   (4)  228  (3)
KMC11                            00000D34-R  271   (4)  228  (3)
KW11K                            00000D66-R  272   (4)  228  (3)
LA100                            00000DA8-R  274   (4)  228  (3)
LA12                             00000DA0-R  273   (4)  228  (3)
LA120                            00000DB1-R  275   (4)  228  (3)
LA180                            00000DBA-R  276   (4)  228  (3)
LA34                             00000DC3-R  277   (4)  228  (3)
LA36                             00000DCB-R  278   (4)  228  (3)
LA38                             00000DD3-R  279   (4)  228  (3)
LESI                             00000DDB-R  280   (4)  228  (3)
LN01                             00000E0B-R  281   (4)  228  (3)
LP04                             00000E13-R  282   (4)  228  (3)
LP05                             00000E1B-R  283   (4)  228  (3)
LP06                             00000E23-R  284   (4)  228  (3)
LP07                             00000E2B-R  285   (4)  228  (3)
LP11                             00000E33-R  286   (4)  228  (3)
LP14                             00000E64-R  287   (4)  228  (3)
LP25                             00000E6C-R  288   (4)  228  (3)
LP26                             00000E74-R  289   (4)  228  (3)
LP27                             00000E7C-R  290   (4)  228  (3)
LPA11K                           00000E84-R  291   (4)  228  (3)
MA780                            00000EB7-R  292   (4)  228  (3)
MBE                              00000EF3-R  293   (4)  228  (3)
ML11                             00000EFA-R  294   (4)  228  (3)
MS750                            00000F36-R  295   (4)  228  (3)
MS780                            00000F3F-R  296   (4)  228  (3)
PCL11                            00000F6E-R  297   (4)  228  (3)
PX780                            00000FA0-R  298   (4)  228  (3)
R80                              00000FCF-R  299   (4)  228  (3)
RA60                             00000FD6-R  300   (4)  228  (3)
RA80                             00000FDE-R  301   (4)  228  (3)
RA81                             00000FE6-R  302   (4)  228  (3)
RB730                            00000FEE-R  303   (4)  228  (3)
```

RC25	00001000-R	306	(4)	228	(3)
RCF25	00000FF7-R	304	(4)	228	(3)
RD51	00001008-R	307	(4)	228	(3)
RD52	00001010-R	308	(4)	228	(3)
RH750	00001018-R	309	(4)	228	(3)
RH780	0000102D-R	310	(4)	228	(3)
RK06	0000108F-R	314	(4)	228	(3)
RK07	00001097-R	315	(4)	228	(3)
RK611	0000109F-R	316	(4)	228	(3)
RL01	0000104E-R	311	(4)	228	(3)
RL02	00001056-R	312	(4)	228	(3)
RL11	0000105E-R	313	(4)	228	(3)
RM03	000010D1-R	317	(4)	228	(3)
RM05	000010D9-R	318	(4)	228	(3)
RM80	000010E1-R	319	(4)	228	(3)
RP04	000010E9-R	320	(4)	228	(3)
RP05	000010F1-R	321	(4)	228	(3)
RP06	000010F9-R	322	(4)	228	(3)
RP07	00001101-R	323	(4)	228	(3)
RQDX1	00001109-R	324	(4)	228	(3)
RX02	0000111E-R	325	(4)	228	(3)
RX211	00001126-R	326	(4)	228	(3)
RX50	00001158-R	327	(4)	228	(3)
SBIA	00001160-R	328	(4)	228	(3)
TE16	00001168-R	329	(4)	228	(3)
TM03	00001170-R	330	(4)	228	(3)
TM78	00001187-R	331	(4)	228	(3)
TS04	0000119E-R	332	(4)	228	(3)
TS05	000011CF-R	333	(4)	228	(3)
TS11	00001218-R	334	(4)	228	(3)
TU45	00001249-R	335	(4)	228	(3)
TU58	00001251-R	336	(4)	228	(3)
TU77	00001259-R	337	(4)	228	(3)
TU78	00001261-R	338	(4)	228	(3)
TU80	00001269-R	339	(4)	228	(3)
TU81	00001282-R	340	(4)	228	(3)
UBE	000012E3-R	341	(4)	228	(3)
UDA50	00001307-R	342	(4)	228	(3)
UNA11	0000134F-R	343	(4)	228	(3)
VS100	00001381-R	344	(4)	228	(3)
VS125	000013B3-R	345	(4)	228	(3)
VS300	000013E5-R	346	(4)	228	(3)
VT100	0000142F-R	350	(4)	228	(3)
VT220	00001438-R	351	(4)	228	(3)
VT240	00001441-R	352	(4)	228	(3)
VT50	00001417-R	347	(4)	228	(3)
VT52	0000141F-R	348	(4)	228	(3)
VT55	00001427-R	349	(4)	228	(3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	118 (2)	118 (2) 232 (4) 233 (4) 237 (4) 238 (4) 239 (4) 240 (4) 241 (4) 242 (4) 243 (4) 244 (4) 245 (4) 246 (4) 247 (4) 248 (4) 249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 265 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 286 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4) 293 (4) 294 (4) 295 (4) 296 (4) 297 (4) 298 (4) 299 (4) 300 (4) 301 (4) 302 (4) 303 (4) 304 (4) 306 (4) 307 (4) 308 (4) 309 (4) 310 (4) 311 (4) 312 (4) 313 (4) 314 (4) 315 (4) 316 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 322 (4) 323 (4) 324 (4) 325 (4) 326 (4) 327 (4) 328 (4) 329 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 335 (4) 336 (4) 337 (4) 338 (4) 339 (4) 340 (4) 341 (4) 342 (4) 343 (4) 344 (4) 345 (4) 346 (4) 347 (4) 348 (4) 349 (4) 350 (4) 351 (4) 352 (4)
$DS_$ADD	1	181 (2)	234 (4) 259 (4) 328 (4)
$DS_$CASE	1	184 (2)	234 (4) 259 (4)
$DS_$COMPLEMENT	1	178 (2)	232 (4) 233 (4) 234 (4) 235 (4) 236 (4) 238 (4) 241 (4) 243 (4) 244 (4) 245 (4) 246 (4) 247 (4) 248 (4) 249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 260 (4) 261 (4) 264 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 280 (4) 286 (4) 291 (4) 292 (4) 294 (4) 296 (4) 297 (4) 298 (4) 309 (4) 310 (4) 313 (4) 316 (4) 326 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 339 (4) 340 (4) 342 (4) 343 (4) 344 (4)
$DS_$DECIMAL	1	133 (2)	345 (4) 346 (4)
$DS_$END	1	129 (2)	232 (4) 233 (4) 234 (4) 235 (4) 236 (4) 237 (4) 238 (4) 239 (4) 240 (4) 241 (4) 242 (4) 243 (4) 244 (4) 245 (4) 246 (4) 247 (4) 248 (4) 249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 286 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4)

				293 (4) 294 (4) 295 (4) 296 (4) 297 (4)
				298 (4) 299 (4) 300 (4) 301 (4) 302 (4)
				303 (4) 304 (4) 306 (4) 307 (4) 308 (4)
				309 (4) 310 (4) 311 (4) 312 (4) 313 (4)
				314 (4) 315 (4) 316 (4) 317 (4) 318 (4)
				319 (4) 320 (4) 321 (4) 322 (4) 323 (4)
				324 (4) 325 (4) 326 (4) 327 (4) 328 (4)
				329 (4) 330 (4) 331 (4) 332 (4) 333 (4)
				334 (4) 335 (4) 336 (4) 337 (4) 338 (4)
				339 (4) 340 (4) 341 (4) 342 (4) 343 (4)
				344 (4) 345 (4) 346 (4) 347 (4) 348 (4)
				349 (4) 350 (4) 351 (4) 352 (4)
$DS_$FETCH	1	172	(2)	234 (4) 235 (4) 236 (4) 259 (4) 260 (4)
				293 (4) 294 (4) 309 (4) 310 (4) 317 (4)
				318 (4) 319 (4) 320 (4) 321 (4) 322 (4)
				323 (4) 328 (4)
$DS_$HEX	1	145	(2)	266 (4) 267 (4) 268 (4) 269 (4) 270 (4)
$DS_$INITIALIZE	1	120	(2)	232 (4) 233 (4) 234 (4) 235 (4) 236 (4)
				237 (4) 238 (4) 239 (4) 240 (4) 241 (4)
				242 (4) 243 (4) 244 (4) 245 (4) 246 (4)
				247 (4) 248 (4) 249 (4) 250 (4) 251 (4)
				252 (4) 253 (4) 254 (4) 255 (4) 256 (4)
				257 (4) 258 (4) 259 (4) 260 (4) 261 (4)
				262 (4) 263 (4) 264 (4) 266 (4) 267 (4)
				268 (4) 269 (4) 270 (4) 271 (4) 272 (4)
				273 (4) 274 (4) 275 (4) 276 (4) 277 (4)
				278 (4) 279 (4) 280 (4) 281 (4) 282 (4)
				283 (4) 284 (4) 285 (4) 286 (4) 287 (4)
				288 (4) 289 (4) 290 (4) 291 (4) 292 (4)
				293 (4) 294 (4) 295 (4) 296 (4) 297 (4)
				298 (4) 299 (4) 300 (4) 301 (4) 302 (4)
				303 (4) 304 (4) 306 (4) 307 (4) 308 (4)
				309 (4) 310 (4) 311 (4) 312 (4) 313 (4)
				314 (4) 315 (4) 316 (4) 317 (4) 318 (4)
				319 (4) 320 (4) 321 (4) 322 (4) 323 (4)
				324 (4) 325 (4) 326 (4) 327 (4) 328 (4)
				329 (4) 330 (4) 331 (4) 332 (4) 333 (4)
				334 (4) 335 (4) 336 (4) 337 (4) 338 (4)
				339 (4) 340 (4) 341 (4) 342 (4) 343 (4)
				344 (4) 345 (4) 346 (4) 347 (4) 348 (4)
				349 (4) 350 (4) 351 (4) 352 (4)
$DS_$LITERAL	1	168	(2)	234 (4) 235 (4) 236 (4) 238 (4) 244 (4)
				246 (4) 248 (4) 250 (4) 254 (4) 255 (4)
				258 (4) 259 (4) 260 (4) 266 (4) 267 (4)
				268 (4) 269 (4) 270 (4) 273 (4) 274 (4)
				275 (4) 276 (4) 277 (4) 278 (4) 279 (4)
				281 (4) 282 (4) 283 (4) 284 (4) 285 (4)
				287 (4) 288 (4) 289 (4) 290 (4) 291 (4)
				292 (4) 294 (4) 296 (4) 298 (4) 299 (4)
				300 (4) 301 (4) 302 (4) 303 (4) 304 (4)
				306 (4) 307 (4) 308 (4) 309 (4) 310 (4)
				311 (4) 312 (4) 314 (4) 315 (4) 317 (4)
				318 (4) 319 (4) 320 (4) 321 (4) 322 (4)
				323 (4) 325 (4) 327 (4) 328 (4) 329 (4)
				332 (4) 333 (4) 334 (4) 335 (4) 336 (4)
				337 (4) 338 (4) 339 (4) 340 (4) 344 (4)
				345 (4) 346 (4) 347 (4) 348 (4) 349 (4)

$DS_$LOGICAL 1 160 (2) 350 (4) 351 (4) 352 (4)
242 (4) 245 (4) 250 (4) 254 (4) 255 (4)
266 (4) 267 (4) 268 (4) 269 (4) 270 (4)
$DS_$NAME 1 126 (2) 232 (4) 233 (4) 234 (4) 235 (4) 236 (4)
238 (4) 239 (4) 240 (4) 241 (4) 242 (4)
243 (4) 245 (4) 246 (4) 247 (4) 248 (4)
249 (4) 250 (4) 251 (4) 252 (4) 253 (4)
254 (4) 255 (4) 256 (4) 257 (4) 258 (4)
259 (4) 260 (4) 261 (4) 262 (4) 263 (4)
264 (4) 266 (4) 267 (4) 268 (4) 269 (4)
270 (4) 271 (4) 272 (4) 273 (4) 274 (4)
275 (4) 276 (4) 277 (4) 278 (4) 279 (4)
280 (4) 281 (4) 282 (4) 283 (4) 284 (4)
285 (4) 286 (4) 287 (4) 288 (4) 289 (4)
290 (4) 291 (4) 292 (4) 294 (4) 295 (4)
296 (4) 298 (4) 299 (4) 300 (4) 301 (4)
302 (4) 303 (4) 304 (4) 306 (4) 307 (4)
308 (4) 309 (4) 310 (4) 311 (4) 312 (4)
313 (4) 314 (4) 315 (4) 316 (4) 317 (4)
318 (4) 319 (4) 320 (4) 321 (4) 322 (4)
323 (4) 324 (4) 325 (4) 326 (4) 327 (4)
328 (4) 329 (4) 330 (4) 331 (4) 332 (4)
333 (4) 334 (4) 335 (4) 336 (4) 337 (4)
338 (4) 339 (4) 340 (4) 341 (4) 342 (4)
343 (4) 344 (4) 345 (4) 346 (4) 347 (4)
348 (4) 349 (4) 350 (4) 351 (4) 352 (4)
$DS_$OCTAL 1 139 (2) 232 (4) 233 (4) 238 (4) 239 (4) 241 (4)
242 (4) 243 (4) 244 (4) 245 (4) 246 (4)
247 (4) 248 (4) 249 (4) 250 (4) 251 (4)
252 (4) 253 (4) 256 (4) 257 (4) 261 (4)
262 (4) 263 (4) 264 (4) 271 (4) 272 (4)
280 (4) 286 (4) 291 (4) 297 (4) 313 (4)
316 (4) 324 (4) 326 (4) 332 (4) 333 (4)
334 (4) 339 (4) 340 (4) 341 (4) 342 (4)
343 (4) 344 (4) 345 (4) 346 (4)
$DS_$STORE 1 175 (2) 232 (4) 233 (4) 234 (4) 235 (4) 236 (4)
238 (4) 239 (4) 241 (4) 242 (4) 243 (4)
244 (4) 245 (4) 246 (4) 247 (4) 248 (4)
249 (4) 250 (4) 251 (4) 252 (4) 253 (4)
254 (4) 255 (4) 256 (4) 257 (4) 258 (4)
259 (4) 260 (4) 261 (4) 262 (4) 263 (4)
264 (4) 266 (4) 267 (4) 268 (4) 269 (4)
270 (4) 271 (4) 272 (4) 273 (4) 274 (4)
275 (4) 276 (4) 277 (4) 278 (4) 279 (4)
280 (4) 281 (4) 282 (4) 283 (4) 284 (4)
285 (4) 286 (4) 287 (4) 288 (4) 289 (4)
290 (4) 291 (4) 292 (4) 293 (4) 294 (4)
296 (4) 297 (4) 298 (4) 299 (4) 300 (4)
301 (4) 302 (4) 303 (4) 304 (4) 306 (4)
307 (4) 308 (4) 309 (4) 310 (4) 311 (4)
312 (4) 313 (4) 314 (4) 315 (4) 316 (4)
317 (4) 318 (4) 319 (4) 320 (4) 321 (4)
322 (4) 323 (4) 324 (4) 325 (4) 326 (4)
327 (4) 328 (4) 329 (4) 330 (4) 331 (4)
332 (4) 333 (4) 334 (4) 335 (4) 336 (4)
337 (4) 338 (4) 339 (4) 340 (4) 341 (4)
342 (4) 343 (4) 344 (4) 345 (4) 346 (4)

				347	(4)	348	(4)	349	(4)	350	(4)	351	(4)		
				352	(4)										
$DS_$STRING	1	151	(2)	234	(4)	242	(4)	245	(4)	246	(4)	247	(4)		
				248	(4)	250	(4)	254	(4)	255	(4)	261	(4)		
				294	(4)										
$DS_AA11K	1	232	(4)	232	(4)										
$DS_AA11K_DEF	1	232	(4)	232	(4)										
$DS_AD11K	1	233	(4)	233	(4)										
$DS_AD11K_DEF	1	233	(4)	233	(4)										
$DS_CI750	2	235	(4)	235	(4)										
$DS_CI780	3	236	(4)	236	(4)										
$DS_CI_NODE	4	234	(4)	234	(4)										
$DS_CONSOLE	1	237	(4)	237	(4)										
$DS_CONSOLE_DEF	1	237	(4)	237	(4)										
$DS_CR11	1	238	(4)	238	(4)										
$DS_CR11_DEF	1	238	(4)	238	(4)										
$DS_DEQNA	1	239	(4)	239	(4)										
$DS_DEQNA_DEF	1	239	(4)	239	(4)										
$DS_DEVTYP	1	187	(2)	206	(3)										
$DS_DISK	1	240	(4)	240	(4)										
$DS_DISK_DEF	1	240	(4)	240	(4)										
$DS_DL11	1	241	(4)	241	(4)										
$DS_DL11_DEF	1	241	(4)	241	(4)										
$DS_DLVJ1	2	242	(4)	242	(4)										
$DS_DLVJ1_DEF	1	242	(4)	242	(4)										
$DS_DMC11	1	243	(4)	243	(4)										
$DS_DMC11_DEF	1	243	(4)	243	(4)										
$DS_DMF32	1	244	(4)	244	(4)										
$DS_DMF32A	2	245	(4)	245	(4)										
$DS_DMF32A_DEF	2	245	(4)	245	(4)										
$DS_DMF32P	1	246	(4)	246	(4)										
$DS_DMF32P_DEF	1	246	(4)	246	(4)										
$DS_DMF32S	1	247	(4)	247	(4)										
$DS_DMF32S_DEF	2	247	(4)	247	(4)										
$DS_DMF32_DEF	1	244	(4)	244	(4)										
$DS_DMP11	2	248	(4)	248	(4)										
$DS_DMP11_DEF	1	248	(4)	248	(4)										
$DS_DMR11	1	249	(4)	249	(4)										
$DS_DMR11_DEF	1	249	(4)	249	(4)										
$DS_DMZ32	3	250	(4)	250	(4)										
$DS_DMZ32_DEF	2	250	(4)	250	(4)										
$DS_DR11B	1	251	(4)	251	(4)										
$DS_DR11B_DEF	1	251	(4)	251	(4)										
$DS_DR11K	1	252	(4)	252	(4)										
$DS_DR11K_DEF	1	252	(4)	252	(4)										
$DS_DR11W	1	253	(4)	253	(4)										
$DS_DR11W_DEF	1	253	(4)	253	(4)										
$DS_DR750	2	254	(4)	254	(4)										
$DS_DR750_DEF	1	254	(4)	254	(4)										
$DS_DR780	2	255	(4)	255	(4)										
$DS_DR780_DEF	1	255	(4)	255	(4)										
$DS_DUP11	1	256	(4)	256	(4)										
$DS_DUP11_DEF	1	256	(4)	256	(4)										
$DS_DV11	1	257	(4)	257	(4)										
$DS_DV11_DEF	1	257	(4)	257	(4)										
$DS_DW730	1	258	(4)	258	(4)										
$DS_DW730_DEF	1	258	(4)	258	(4)										

$DS_DW750	1	259	(4)	259	(4)
$DS_DW750_DEF	1	259	(4)	259	(4)
$DS_DW780	2	260	(4)	260	(4)
$DS_DW780_DEF	1	260	(4)	260	(4)
$DS_DZ11	1	261	(4)	261	(4)
$DS_DZ11_DEF	1	261	(4)	261	(4)
$DS_DZ32	2	262	(4)	262	(4)
$DS_DZ32_DEF	1	262	(4)	262	(4)
$DS_DZV11	1	263	(4)	263	(4)
$DS_DZV11_DEF	1	263	(4)	263	(4)
$DS_HPODEF	2	118	(2)	118	(2)
$DS_IEU11A	1	264	(4)	264	(4)
$DS_IEU11A_DEF	1	264	(4)	264	(4)
$DS_KA730	3	266	(4)	266	(4)
$DS_KA750	3	267	(4)	267	(4)
$DS_KA780	2	268	(4)	268	(4)
$DS_KA785	2	270	(4)	270	(4)
$DS_KAXXX	2	269	(4)	269	(4)
$DS_KA_DEF	3	265	(4)	265	(4)
$DS_KMC11	1	271	(4)	271	(4)
$DS_KMC11_DEF	1	271	(4)	271	(4)
$DS_KW11K	1	272	(4)	272	(4)
$DS_KW11K_DEF	1	272	(4)	272	(4)
$DS_LA100	1	274	(4)	274	(4)
$DS_LA100_DEF	1	274	(4)	274	(4)
$DS_LA12	1	273	(4)	273	(4)
$DS_LA120	1	275	(4)	275	(4)
$DS_LA120_DEF	1	275	(4)	275	(4)
$DS_LA12_DEF	1	273	(4)	273	(4)
$DS_LA180	1	276	(4)	276	(4)
$DS_LA180_DEF	1	276	(4)	276	(4)
$DS_LA34	1	277	(4)	277	(4)
$DS_LA34_DEF	1	277	(4)	277	(4)
$DS_LA36	1	278	(4)	278	(4)
$DS_LA36_DEF	1	278	(4)	278	(4)
$DS_LA38	1	279	(4)	279	(4)
$DS_LA38_DEF	1	279	(4)	279	(4)
$DS_LESI	1	280	(4)	280	(4)
$DS_LESI_DEF	1	280	(4)	280	(4)
$DS_LN01	1	281	(4)	281	(4)
$DS_LN01_DEF	1	281	(4)	281	(4)
$DS_LP04	1	282	(4)	282	(4)
$DS_LP04_DEF	1	282	(4)	282	(4)
$DS_LP05	1	283	(4)	283	(4)
$DS_LP05_DEF	1	283	(4)	283	(4)
$DS_LP06	1	284	(4)	284	(4)
$DS_LP06_DEF	1	284	(4)	284	(4)
$DS_LP07	1	285	(4)	285	(4)
$DS_LP07_DEF	1	285	(4)	285	(4)
$DS_LP11	1	286	(4)	286	(4)
$DS_LP11_DEF	1	286	(4)	286	(4)
$DS_LP14	1	287	(4)	287	(4)
$DS_LP14_DEF	1	287	(4)	287	(4)
$DS_LP25	1	288	(4)	288	(4)
$DS_LP25_DEF	1	288	(4)	288	(4)
$DS_LP26	1	289	(4)	289	(4)
$DS_LP26_DEF	1	289	(4)	289	(4)

$DS_LP27	1	290	(4)	290	(4)
$DS_LP27_DEF	1	290	(4)	290	(4)
$DS_LPA11K	1	291	(4)	291	(4)
$DS_LPA11K_DEF	1	291	(4)	291	(4)
$DS_MA780	2	292	(4)	292	(4)
$DS_MA780_DEF	1	292	(4)	292	(4)
$DS_MBE	1	293	(4)	293	(4)
$DS_MBE_DEF	1	293	(4)	293	(4)
$DS_ML11	1	294	(4)	294	(4)
$DS_ML11_DEF	1	294	(4)	294	(4)
$DS_MS750	1	295	(4)	295	(4)
$DS_MS750_DEF	1	295	(4)	295	(4)
$DS_MS780	1	296	(4)	296	(4)
$DS_MS780_DEF	1	296	(4)	296	(4)
$DS_PCL11	1	297	(4)	297	(4)
$DS_PCL11_DEF	1	297	(4)	297	(4)
$DS_PX780	2	298	(4)	298	(4)
$DS_PX780_DEF	1	298	(4)	298	(4)
$DS_R80	1	299	(4)	299	(4)
$DS_R80_DEF	1	299	(4)	299	(4)
$DS_RA60	1	300	(4)	300	(4)
$DS_RA60_DEF	1	300	(4)	300	(4)
$DS_RA80	1	301	(4)	301	(4)
$DS_RA80_DEF	1	301	(4)	301	(4)
$DS_RA81	1	302	(4)	302	(4)
$DS_RA81_DEF	1	302	(4)	302	(4)
$DS_RB730	2	303	(4)	303	(4)
$DS_RB730_DEF	1	303	(4)	303	(4)
$DS_RC25	1	306	(4)	306	(4)
$DS_RC25_DEF	1	306	(4)	306	(4)
$DS_RCF25	1	304	(4)	304	(4)
$DS_RCF25_DEF	1	304	(4)	304	(4)
$DS_RD51	1	307	(4)	307	(4)
$DS_RD51_DEF	1	307	(4)	307	(4)
$DS_RD52	1	308	(4)	308	(4)
$DS_RD52_DEF	1	308	(4)	308	(4)
$DS_RH750	2	309	(4)	309	(4)
$DS_RH750_DEF	1	309	(4)	309	(4)
$DS_RH780	2	310	(4)	310	(4)
$DS_RH780_DEF	1	310	(4)	310	(4)
$DS_RK06	1	314	(4)	314	(4)
$DS_RK06_DEF	1	314	(4)	314	(4)
$DS_RK07	1	315	(4)	315	(4)
$DS_RK07_DEF	1	315	(4)	315	(4)
$DS_RK611	1	316	(4)	316	(4)
$DS_RK611_DEF	1	316	(4)	316	(4)
$DS_RL01	1	311	(4)	311	(4)
$DS_RL01_DEF	1	311	(4)	311	(4)
$DS_RL02	1	312	(4)	312	(4)
$DS_RL02_DEF	1	312	(4)	312	(4)
$DS_RL11	1	313	(4)	313	(4)
$DS_RL11_DEF	1	313	(4)	313	(4)
$DS_RM03	1	317	(4)	317	(4)
$DS_RM03_DEF	1	317	(4)	317	(4)
$DS_RM05	1	318	(4)	318	(4)
$DS_RM05_DEF	1	318	(4)	318	(4)
$DS_RM80	1	319	(4)	319	(4)

$DS_RM80_DEF	1	319	(4)	319	(4)
$DS_RP04	1	320	(4)	320	(4)
$DS_RP04_DEF	1	320	(4)	320	(4)
$DS_RP05	1	321	(4)	321	(4)
$DS_RP05_DEF	1	321	(4)	321	(4)
$DS_RP06	1	322	(4)	322	(4)
$DS_RP06_DEF	1	322	(4)	322	(4)
$DS_RP07	1	323	(4)	323	(4)
$DS_RP07_DEF	1	323	(4)	323	(4)
$DS_RQDXT	1	324	(4)	324	(4)
$DS_RQDX1_DEF	1	324	(4)	324	(4)
$DS_RX02	1	325	(4)	325	(4)
$DS_RX02_DEF	1	325	(4)	325	(4)
$DS_RX21T	1	326	(4)	326	(4)
$DS_RX211_DEF	1	326	(4)	326	(4)
$DS_RX50	1	327	(4)	327	(4)
$DS_RX50_DEF	1	327	(4)	327	(4)
$DS_SBIA	2	328	(4)	328	(4)
$DS_SBIA_DEF	1	328	(4)	328	(4)
$DS_TE16	1	329	(4)	329	(4)
$DS_TE16_DEF	1	329	(4)	329	(4)
$DS_TM03	1	330	(4)	330	(4)
$DS_TM03_DEF	1	330	(4)	330	(4)
$DS_TM78	1	331	(4)	331	(4)
$DS_TM78_DEF	1	331	(4)	331	(4)
$DS_TS04	1	332	(4)	332	(4)
$DS_TS04_DEF	1	332	(4)	332	(4)
$DS_TS05	2	333	(4)	333	(4)
$DS_TS05_DEF	1	333	(4)	333	(4)
$DS_TS11	1	334	(4)	334	(4)
$DS_TS11_DEF	1	334	(4)	334	(4)
$DS_TU45	1	335	(4)	335	(4)
$DS_TU45_DEF	1	335	(4)	335	(4)
$DS_TU58	1	336	(4)	336	(4)
$DS_TU58_DEF	1	336	(4)	336	(4)
$DS_TU77	1	337	(4)	337	(4)
$DS_TU77_DEF	1	337	(4)	337	(4)
$DS_TU78	1	338	(4)	338	(4)
$DS_TU78_DEF	1	338	(4)	338	(4)
$DS_TU80	2	339	(4)	339	(4)
$DS_TU80_DEF	1	339	(4)	339	(4)
$DS_TU81	2	340	(4)	340	(4)
$DS_TU81_DEF	1	340	(4)	340	(4)
$DS_UBE	1	341	(4)	341	(4)
$DS_UBE_DEF	1	341	(4)	341	(4)
$DS_UDA50	1	342	(4)	342	(4)
$DS_UDA50_DEF	1	342	(4)	342	(4)
$DS_UNA11	2	343	(4)	343	(4)
$DS_UNA11_DEF	1	343	(4)	343	(4)
$DS_VS100	1	344	(4)	344	(4)
$DS_VS100_DEF	1	344	(4)	344	(4)
$DS_VS125	1	345	(4)	345	(4)
$DS_VS125_DEF	1	345	(4)	345	(4)
$DS_VS300	1	346	(4)	346	(4)
$DS_VS300_DEF	1	346	(4)	346	(4)
$DS_VT100	1	350	(4)	350	(4)
$DS_VT100_DEF	1	350	(4)	350	(4)

$DS_VT220	1	351	(4)	351	(4)
$DS_VT220_DEF	1	351	(4)	351	(4)
$DS_VT240	1	352	(4)	352	(4)
$DS_VT240_DEF	1	352	(4)	352	(4)
$DS_VT50	1	347	(4)	347	(4)
$DS_VT50_DEF	1	347	(4)	347	(4)
$DS_VT52	1	348	(4)	348	(4)
$DS_VT52_DEF	1	348	(4)	348	(4)
$DS_VT55	1	349	(4)	349	(4)
$DS_VT55_DEF	1	349	(4)	349	(4)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.11	00:00:00.53
Command processing	120	00:00:00.73	00:00:02.27
Pass 1	2806	00:00:36.54	00:01:00.47
Symbol table sort	3	00:00:00.74	00:00:01.44
Pass 2	1507	00:00:08.19	00:00:15.25
Symbol table output	64	00:00:00.38	00:00:00.40
Psect synopsis output	7	00:00:00.02	00:00:00.02
Cross-reference output	483	00:00:02.41	00:00:03.40
Assembler run totals	5028	00:00:49.15	00:01:23.80

The working set limit was 1000 pages.
450239 bytes (880 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 555 non-local and 0 local symbols.
353 source lines were read in Pass 1, producing 0 object records in Pass 2.
742 pages of virtual memory were used to define 367 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	232
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	236

2460 GETS were required to define 236 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ICODE/UPDA=(ICODE.UPD,ICODE.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

Table of contents

(2) 161 DECLARATIONS
(5) 222 Device description database

```
0000      1          .TITLE  IOBASE  *** IOBASE I/O data base
0000      2          .ident  /12-46/
0000      3          .LIST   MEB
0000      4          .DSABL  GBL
0000      5          .NLIST  CND
0000      6
0000      7 ;
0000      8 ;
0000      9 ; Copyright (c) 1977, 1982, 1983
0000     10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000     11 ;
0000     12 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
0000     13 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
0000     14 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
0000     15 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
0000     16 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
0000     17 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
0000     18 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
0000     19 ;
0000     20 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
0000     21 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
0000     22 ; EQUIPMENT CORPORATION.
0000     23 ;
0000     24 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000     25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     26 ;
0000     27 ; R.HEINEN 3-AUG-76
0000     28 ;
0000     29 ; MODIFICATION HISTORY:
0000     30 ;
0000     31 ;                 N. HOWGATE 3-FEB-78     VERSION 02 (ESSAA-4.00).
0000     32 ;          01     ALTERED TO REPRESENT A ONE (4) DBA: SYSTEM
0000     33 ;                 FOR INITIAL DIAGNOSTIC SUPERVISOR QIO DEBUG
0000     34 ;
0000     35 ;                 N. HOWGATE  20-APR-78  VERSION 03  (ESSAA-4.01).
0000     36 ;          02     ADDED DEVICE DATA FOR ONE DMA: SYSTEM (3 UNITS)
0000     37 ;
0000     38 ;                 TOM SOUTTER  12-MAY-78  VERSION 04  (ESSAA-4.02).
0000     39 ;          03     ADDED DEVICE DATA FOR ONE DRA: SYSTEM (4 UNITS)
0000     40 ;
0000     41 ;                 TOM SOUTTER  14-JUN-78  VERSION 05  (ESSAA-4.03).
0000     42 ;          04     ADDED DEVICE DATA FOR ONE MTA: SYSTEM (2 UNITS)
0000     43 ;                 NICK HOWGATE     22-JUN-78
0000     44 ;          05     ADD LONGWORD TO END OF RM03 UCB'S FOR ADDITIONAL STATUS
0000     45 ;                 NICK HOWGATE     11-JUL-78
0000     46 ;          06     ADD RK07 SUPPORT IN DM TYPE UCB
0000     47 ;                 NICK HOWGATE     13-JUL-78
0000     48 ;          07     ADD LONGWORD TO CRB FOR UNIT INIT CODE
0000     49 ;                 CRB$L_INTD+VEC$L_UNITINIT
0000     50 ;                 NICK HOWGATE     24-AUG-78
0000     51 ;          08     INCLUDE NEW UCB ENTRIES (UCB$W_ERRCNT ,UCB$B_FEX, UCB$B_CEX)
0000     52 ;                 Roger Riggs      30-SEP-78       VERSION 6 (ESSAA 5.00)
0000     53 ;          09     Major modifications to accomodate new QIO initialization
0000     54 ;                 and loadable drivers.
0000     55 ;                 Roger Riggs      1-OCT-1979
0000     56 ;          10     Added DMP11,DMR11,DV11,LA180,MS780
0000     57 ;          11     John Ciukaj      1-15-80
```

```
0000     58 ;                  Added DL11,RX02,RX211,TU58,TS11
0000     59 ;
0000     60 ;          12      M. Baggett      18-MAR-80
0000     61 ;                  Add TU78 magtape
0000     62 ;                  Dave Butenhof, 15-dec-1980, V6.2
0000     63 ;          13      Add ML11 PTDESC.
0000     64 ;                  Dave Butenhof, 19-mar-1981, version 6.3
0000     65 ;          14      Add DV11 PTdesc.
0000     66 ;          15      Add KA730 Ptdesc.
0000     67 ;          16      -dave butenhof, 4-sep-1981, version 6.5
0000     68 ;                  add RB730 and R80 Ptable descriptors
0000     69 ;
0000     70 ;          17      - Dave Butenhof, 26-Oct-1981, version 6.-
0000     71 ;                  Add some missing devices
0000     72 ;
0000     73 ;          18      - Dave Butenhof, 10-Nov-1981, version 6.-
0000     74 ;                  Add the UDA50 and RA80 devices
0000     75 ;
0000     76 ;          19      - Dave Butenhof, 15-Dec-1981, Version 6.6
0000     77 ;                  Add PX780
0000     78 ;
0000     79 ;          20      - Dave Butenhof, 23-Feb-1982, version 6.6
0000     80 ;                  Add CI_NODE and CI750 Ptable descriptors
0000     81 ;          [21] Dave Butenhof, 16-Apr-1982, verison 6.7
0000     82 ;                  Add RA81 Ptable descriptor
0000     83 ;          [22] Dave Butenhof, 29-Apr-1982, version 6.7
0000     84 ;                  Add RA60 Ptable descriptor
0000     85 ;          [23] Dave Butenhof, 05-May-1982, version 6.8
0000     86 ;                  Activate the device name check feature which
0000     87 ;                  was added (latent) in 6.7 macros.
0000     88 ;          [24] Dave Butenhof, 11-May-1982, version 6.8
0000     89 ;                  Add LP07 and LP26 Ptables for Dan Milleville
0000     90 ;          [25] Dave Butenhof, 25-May-1982, version 6.8
0000     91 ;                  Add "console" Ptable desc. to IOBASE.  This allows
0000     92 ;                  new console Ptable to be displayed by SHOW
0000     93 ;                  {DEVICE ! SELECT} without the annoying "..." which
0000     94 ;                  means it couldn't find the Ptable.  This device is
0000     95 ;                  attached at DS startup by KERNEL, and as it has
0000     96 ;                  a lower-case device type, CANNOT be attached manually!
0000     97 ;          [26] M. Baggett, 20-Jul-1982 Ver 6.9
0000     98 ;                  Added LP27 line printer.
0000     99 ;
0000    100 ;          27      Bob Bergazzi    1-Oct-82        Version 6.9
0000    101 ;                  Added UNA11 (UNIBUS to NI Adapter)
0000    102 ;
0000    103 ;          28      M. Baggett      1-Dec-1982      Version 6.10
0000    104 ;                  Added TU80 magtape and DZ32.
0000    105 ;
0000    106 ;          29      M. Baggett      8-Dec-1982      Version 6.10
0000    107 ;                  Added TU81 magtape.
0000    108 ;
0000    109 ;          30      M. Baggett      10-Dec-1982     Version 6.10
0000    110 ;                  Added VS100.
0000    111 ;
0000    112 ;          31      Bob Bergazzi    10-Jan-83       Version 6.11
0000    113 ;                  Changed the comment before DS$GA_PTABLE.
0000    114 ;
```

```
0000   115 ;
0000   116 ;       32      M. Baggett      14-Jan-83       Version 6.11
0000   117 ;               Added RC25.
0000   118 ;       33      M. Baggett      15-Feb-83       Version 6.11
0000   119 ;               Added VS300 and RCF25.
0000   120 ;
0000   121 ;       34      M. Baggett      14-Mar-1983     Version 6.11
0000   122 ;               Added p-table for LESI controller.
0000   123 ;
0000   124 ;       35      M. Baggett      21-March-1983   Version 6.11
0000   125 ;               Added p-tables for LA12,LA100,LN01,VT220,VT240.
0000   126 ;
0000   127 ;       36      Bob Bergazzi    25-April-1983   Version 6.11
0000   128 ;               Added Ptable for KA785.
0000   129 ;
0000   130 ;       37      Richard Brown   2-May-1983      Version 6.11
0000   131 ;               Added SBIA and KAxxx.
0000   132 ;
0000   133 ;       38      Richard Brown   15-July-1983    Version 6-12
0000   134 ;               Added RC11.
0000   135 ;
0000   136 ;       39      M. Baggett      8-Aug-1983      Version 6.12
0000   137 ;               Added IEU11A
0000   138 ;
0000   139 ;       40      M. Baggett      7-Sep-1983      Version 6.13
0000   140 ;               Added VS125.
0000   141 ;
0000   142 ;       41      M. Baggett      21-Sep-1983     Version 6.13
0000   143 ;               Added DISK p-table.
0000   144 ;
0000   145 ;       42      M. Baggett      11-Oct-1983     Version 6.13
0000   146 ;               Added TAPE p-table.
0000   147 ;
0000   148 ;       43      M. Baggett      13-Oct-1983     Version 6.13
0000   149 ;               Added TS05 p-table.
0000   150 ;
0000   151 ;       44      Bob Bergazzi    Jan 23, 1984    Version 6.14
0000   152 ;               Added KAZZZ Ptable.
0000   153 ;
0000   154 ;       45      M. Baggett      Feb 6, 1984     Version 6.14
0000   155 ;               Added DMZ32 p-table.
0000   156 ;
0000   157 ;       46      Richard Brown   July 7, 1984    Version 7.0
0000   158 ;               Removed RC11, since it is really LESI.
0000   159 ;--
```

```
                        0000   161          .SBTTL  DECLARATIONS
                        0000   162 ;
                        0000   163 ; MACRO LIBRARY CALLS
                        0000   164 ;
                        0000   165
                        0000   166 .library         /$diag/
                        0000   167
                        0000   168          $DS_HPODEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
                00000008 0000               .IIF    NB,.BLKQ,       .BLKQ   1
                        0008                .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
                0000000A 0008               .IIF    NB,.BLKW,       .BLKW   1
                        000A                .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
                0000000B 000A               .IIF    NB,.BLKB,       .BLKB   1
                        000B                .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
                0000000C 000B               .IIF    NB,.BLKB,       .BLKB   1
                        000C                .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
                00000018 000C               .IIF    NB,.BLKB,       .BLKB   12
                        0018                .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
                0000001C 0018               .IIF    NB,.BLKL,       .BLKL   1
                        001C                .IIF    NB,HP$A_DVA,    HP$A_DVA:
                00000020 001C               .IIF    NB,.BLKL,       .BLKL   1
                        0020                .IIF    NB,HP$A_LINK,   HP$A_LINK:
                00000024 0020               .IIF    NB,.BLKL,       .BLKL   1
                        0024                .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
                00000026 0024               .IIF    NB,.BLKW,       .BLKW   1
                        0026                .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
                00000032 0026               .IIF    NB,.BLKB,       .BLKB   12
                        0032                .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                        0000                .RESTORE
                        0000   169
```

```
                          0000   171
                          0000   172 ;
                          0000   173 ; GLOBAL DATA
                          0000   174 ;
                          0000   175 ; SYSTEM BOOT UCB TABLES
                          0000   176 ;
                          0000   177
                   00000000   178          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                          0000   179 IOC$AL_SYSUCB::                              ; SYSTEM DEVICE UCB TABLE
              00000000    0000   180          .LONG   0                           ; RP06 UCB TABLE
              00000000    0004   181          .LONG   0
                          0008   182
                          0008   183 IOC$GL_DEVLIST::
              00000000    0008   184          .LONG   0                           ; START OF DEVICE LIST
                          000C   185 IOC$GL_ADPLIST::
              00000000    000C   186          .LONG   0                           ; START OF ADAPTER CONTROL BLOCK LIST
                          0010   187 IOC$GL_DPTLIST::                            ; QUEUE OF DRIVER PROLOGUE BLOCKS
    00000010'00000010'    0010   188          .LONG   IOC$GL_DPTLIST,IOC$GL_DPTLIST
```

```
                                00000000   190          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                    0000   191 ;+
                                    0000   192 ; The following table contains a select bit corresonding to the
                                    0000   193 ; entries in DS$GA_PTABLE.
                                    0000   194 ;-
                         00000080   0000   195 MAXPT=128
                                    0000   196 DS$GA_SELECTS::
                         00000014   0000   197          .BLKL   MAXPT/32+1
                                    0014   198
                                    0014   199 ;+
                                    0014   200 ; This table contains the addresses of each P-table actually to be tested.
                                    0014   201 ; It is built at program start time by matching the device types in the
                                    0014   202 ; program DEVTYP list with the type field in each P-table.
                                    0014   203 ; DSP$GEN_PTABLES in DEVICE actually performs this.
                                    C014   204 ;-
                                    0014   205 DS$GA_SELECTED::
                         00000080   0014   206          .LONG   MAXPT
00000000'00000000'00000000'00000000' 0018   207          .LONG   0[MAXPT]
00000000'00000000'00000000'00000000' 0028
00000000'00000000'00000000'00000000' 0038
00000000'00000000'00000000'00000000' 0048
00000000'00000000'00000000'00000000' 0058
00000000'00000000'00000000'00000000' 0068
00000000'00000000'00000000'00000000' 0078
00000000'00000000'00000000'00000000' 0088
00000000'00000000'00000000'00000000' 0098
00000000'00000000'00000000'00000000' 00A8
00000000'00000000'00000000'00000000' 00B8
00000000'00000000'00000000'00000000' 00C8
00000000'00000000'00000000'00000000' 00D8
00000000'00000000'00000000'00000000' 00E8
00000000'00000000'00000000'00000000' 00F8
00000000'00000000'00000000'00000000' 0108
00000000'00000000'00000000'00000000' 0118
00000000'00000000'00000000'00000000' 0128
00000000'00000000'00000000'00000000' 0138
00000000'00000000'00000000'00000000' 0148
00000000'00000000'00000000'00000000' 0158
00000000'00000000'00000000'00000000' 0168
00000000'00000000'00000000'00000000' 0178
00000000'00000000'00000000'00000000' 0188
00000000'00000000'00000000'00000000' 0198
00000000'00000000'00000000'00000000' 01A8
00000000'00000000'00000000'00000000' 01B8
00000000'00000000'00000000'00000000' 01C8
00000000'00000000'00000000'00000000' 01D8
00000000'00000000'00000000'00000000' 01E8
00000000'00000000'00000000'00000000' 01F8
00000000'00000000'00000000'00000000' 0208
                                    0218   208
                                    0218   209 ;+
                                    0218   210 ; This table contains the address of each known device.  It is ordered.
                                    0218   211 ; The most recently attached device has the lowest index in the table because
                                    0218   212 ; entries are allocated from the bottom up. In addition as a byproduct of
                                    0218   213 ; the SELECT command the specified devices are moved to the end of the list.
                                    0218   214 ; This results in the most recently selected device is the last (in a series)
                                    0218   215 ; to be tested.
```

```
                                     0218   216 ;-
                                     0218   217
                                     0218   218 DS$GA_PTABLE::
                          00000080   0218   219          .LONG   MAXPT
00000000'00000000'00000000'00000000' 021C   220          .LONG   0[MAXPT]
00000000'00000000'00000000'00000000' 022C
00000000'00000000'00000000'00000000' 023C
00000000'00000000'00000000'00000000' 024C
00000000'00000000'00000000'00000000' 025C
00000000'00000000'00000000'00000000' 026C
00000000'00000000'00000000'00000000' 027C
00000000'00000000'00000000'00000000' 028C
00000000'00000000'00000000'00000000' 029C
00000000'00000000'00000000'00000000' 02AC
00000000'00000000'00000000'00000000' 02BC
00000000'00000000'00000000'00000000' 02CC
00000000'00000000'00000000'00000000' 02DC
00000000'00000000'00000000'00000000' 02EC
00000000'00000000'00000000'00000000' 02FC
00000000'00000000'00000000'00000000' 030C
00000000'00000000'00000000'00000000' 031C
00000000'00000000'00000000'00000000' 032C
00000000'00000000'00000000'00000000' 033C
00000000'00000000'00000000'00000000' 034C
00000000'00000000'00000000'00000000' 035C
00000000'00000000'00000000'00000000' 036C
00000000'00000000'00000000'00000000' 037C
00000000'00000000'00000000'00000000' 038C
00000000'00000000'00000000'00000000' 039C
00000000'00000000'00000000'00000000' 03AC
00000000'00000000'00000000'00000000' 03BC
00000000'00000000'00000000'00000000' 03CC
00000000'00000000'00000000'00000000' 03DC
00000000'00000000'00000000'00000000' 03EC
00000000'00000000'00000000'00000000' 03FC
00000000'00000000'00000000'00000000' 040C
```

```
                041C   222          .SBTTL  Device description database
           00000000    223          .PSECT  DATA, SHR, NOWRT, NOEXE, BYTE
                0000   224 ;
                0000   225 ;       The following list contains the descriptor for each type
                0000   226 ;       of device the supervisor known about.
                0000   227 ;
                0000   228
                0000   229 DS$GA_PTDESC::
                0000   230          $DS_DEVTYP      <>,<AA11K,AD11K,-                       ;               [20]
                0000   231                          CI780,CI750,CI_NODE,-                   ;               [20]
                0000   232                          Console,CR11,DISK,DL11,-                ;               [41]
                0000   233                          DMC11,DMF32,DMF32A,DMF32P,-
                0000   234                          DMF32S,DMP11,DMR11,DMZ32,DR11B,-        ;               [45]
                0000   235                          DR11K,DR11W,DR750,DR780,DUP11,-
                0000   236                          DW730,DW750,DW780,-
                0000   237                          DZ11,DZ32,IEU11A,KA730,KA750,KA780,KA785,KAXXX,KAZZZ,-; [44]
                0000   238                          KMC11,KW11K,LA12,LA34,LA36,LA38,-       ;               [35]
                0000   239                          LA100,LA120,LA180,LESI,LN01,LP04, -     ;               [34]
                0000   240                          LP05,LP06,LP07, -
                0000   241                          LP11,LP14,LP25,LP26,LP27,-
                0000   242                          LPA11K,MA780,MBE,ML11, -
                0000   243                          MS750,MS780,PCL11,PX780, -
                0000   244                          R80,RA60,RA80,RA81,RB730,RCF25,RC25,-   ;       [38][46]
                0000   245                          RH750,RH780,RL01,RL02,-
                0000   246                          RL11,RK06,RK07,RK611, -
                0000   247                          RM03,RM05,RM80, -
                0000   248                          RP04,RP05,RP06,RP07,RX02,-
                0000   249                          RX211,SBIA,TAPE,TE16,TM03,TM78, -               ;       [42]
                0000   250                          TS04,TS05,TS11,TU45,TU58,-              ;               [43]
                0000   251                          TU77,TU78,TU80,TU81,UBE,UDA50,UNA11,VS100,VS125,-;      [40]
                0000   252                          VS300,VT50,VT52,VT55,VT100,VT220,VT240>         ;       [35]
00000071        0000       AL_DEVTYP:       .LONG   $$N
000001C8'       0004                        .LONG   AA11K
00000219'       0008                        .LONG   AD11K
000002E9'       000C                        .LONG   CI780
0000026A'       0010                        .LONG   CI750
00000370'       0014                        .LONG   CI_NODE
0000042D'       0018                        .LONG   Console
0000043B'       001C                        .LONG   CR11
00000492'       0020                        .LONG   DISK
000004A2'       0024                        .LONG   DL11
000004EF'       0028                        .LONG   DMC11
0000053D'       002C                        .LONG   DMF32
00000590'       0030                        .LONG   DMF32A
000006B2'       0034                        .LONG   DMF32P
0000072C'       0038                        .LONG   DMF32S
000007A0'       003C                        .LONG   DMP11
00000868'       0040                        .LONG   DMR11
000008B6'       0044                        .LONG   DMZ32
00000A14'       0048                        .LONG   DR11B
00000A62'       004C                        .LONG   DR11K
00000AB3'       0050                        .LONG   DR11W
00000B01'       0054                        .LONG   DR750
00000B8C'       0058                        .LONG   DR780
00000C1F'       005C                        .LONG   DUP11
00000C6D'       0060                        .LONG   DW730
00000C9C'       0064                        .LONG   DW750
```

```
00000CE0' 0068          .LONG   DW780
00000D54' 006C          .LONG   DZ11
00000DBD' 0070          .LONG   DZ32
00000EAE' 0074          .LONG   IEU11A
00000EF9' 0078          .LONG   KA730
00000FD4' 007C          .LONG   KA750
000010A0' 0080          .LONG   KA780
00001149' 0084          .LONG   KA785
000011F2' 0088          .LONG   KAXXX
0000129B' 008C          .LONG   KAZZZ
00001344' 0090          .LONG   KMC11
00001392' 0094          .LONG   KW11K
000013E3' 0098          .LONG   LA12
000013FD' 009C          .LONG   LA34
00001417' 00A0          .LONG   LA36
00001431' 00A4          .LONG   LA38
0000144B' 00A8          .LONG   LA100
00001466' 00AC          .LONG   LA120
00001481' 00B0          .LONG   LA180
0000149C' 00B4          .LONG   LESI
000014E8' 00B8          .LONG   LN01
00001502' 00BC          .LONG   LP04
0000151C' 00C0          .LONG   LP05
00001536' 00C4          .LONG   LP06
00001550' 00C8          .LONG   LP07
0000156A' 00CC          .LONG   LP11
000015B7' 00D0          .LONG   LP14
000015D1' 00D4          .LONG   LP25
000015EB' 00D8          .LONG   LP26
00001605' 00DC          .LONG   LP27
0000161F' 00E0          .LONG   LPA11K
00001678' 00E4          .LONG   MA780
000016F3' 00E8          .LONG   MBE
00001707' 00EC          .LONG   ML11
00001769' 00F0          .LONG   MS750
0000177A' 00F4          .LONG   MS780
000017CA' 00F8          .LONG   PCL11
00001813' 00FC          .LONG   PX780
0000187C' 0100          .LONG   R80
00001895' 0104          .LONG   RA60
000018AF' 0108          .LONG   RA80
000018C9' 010C          .LONG   RA81
000018E3' 0110          .LONG   RB730
00001912' 0114          .LONG   RCF25
0000192D' 0118          .LONG   RC25
00001947' 011C          .LONG   RH750
000019A0' 0120          .LONG   RH780
00001A05' 0124          .LONG   RL01
00001A1F' 0128          .LONG   RL02
00001A39' 012C          .LONG   RL11
00001A86' 0130          .LONG   RK06
00001AA0' 0134          .LONG   RK07
00001ABA' 0138          .LONG   RK611
00001B08' 013C          .LONG   RM03
00001B2C' 0140          .LONG   RM05
00001B50' 0144          .LONG   RM80
00001B74' 0148          .LONG   RP04
```

```
00001B98' 014C          .LONG   RP05
00001BBC' 0150          .LONG   RP06
00001BE0' 0154          .LONG   RP07
00001C04' 0158          .LONG   RX02
00001C1E' 015C          .LONG   RX211
00001C6C' 0160          .LONG   SBIA
00001CAE' 0164          .LONG   TAPE
00001CBE' 0168          .LONG   TE16
00001CD8' 016C          .LONG   TM03
00001D06' 0170          .LONG   TM78
00001D34' 0174          .LONG   TS04
00001D8B' 0178          .LONG   TS05
00001DF5' 017C          .LONG   TS11
00001E4C' 0180          .LONG   TU45
00001E66' 0184          .LONG   TU58
00001E80' 0188          .LONG   TU77
00001E9A' 018C          .LONG   TU78
00001EB4' 0190          .LONG   TU80
00001F23' 0194          .LONG   TU81
00001F7A' 0198          .LONG   UBE
00001FB5' 019C          .LONG   UDA50
0000201E' 01A0          .LONG   UNA11
0000206C' 01A4          .LONG   VS100
00002004' 01A8          .LONG   VS125
0000211C' 01AC          .LONG   VS300
00002174' 01B0          .LONG   VT50
0000218E' 01B4          .LONG   VT52
000021A8' 01B8          .LONG   VT55
000021C2' 01BC          .LONG   VT100
000021DD' 01C0          .LONG   VT220
000021F8' 01C4          .LONG   VT240
          01C8    253
```

```
000001C8   255          .PSECT  DATA, SHR, NOWRT, NOEXE, BYTE
    01C8   256          .NLIST  ME,MEB
    01C8   257          .LIST   MC
    01C8   258
    01C8   259 AA11K:   $DS_AA11K
    0219   260 AD11K:   $DS_AD11K
    026A   261          $DS_CI_DEF                                              ;  [20]
    026A   262 CI750:   $DS_CI750                                               ;  [20]
    02E9   263 CI780:   $DS_CI780
    0370   264 CI_NODE: $DS_CI_NODE                                             ;  [20]
    042D   265 Console:
    042D   266          $DS_Console
    043B   267 CR11:    $DS_CR11
    0492   268 DISK:    $DS_DISK                                                ;  [41]
    04A2   269 DL11:    $DS_DL11
    04EF   270 DMC11:   $DS_DMC11
    053D   271 DMF32:   $DS_DMF32
    0590   272 DMF32A:  $DS_DMF32A
    06B2   273 DMF32P:  $DS_DMF32P
    072C   274 DMF32S:  $DS_DMF32S
    07A0   275 DMP11:   $DS_DMP11
    0868   276 DMR11:   $DS_DMR11
    08B6   277 DMZ32:   $DS_DMZ32                                                       ;  [45]
    0A14   278 DR11B:   $DS_DR11B
    0A62   279 DR11K:   $DS_DR11K
    0AB3   280 DR11W:   $DS_DR11W
    0B01   281 DR750:   $DS_DR750
    0B8C   282 DR780:   $DS_DR780
    0C1F   283 DUP11:   $DS_DUP11
    0C6D   284 DW730:   $DS_DW730                                                       ;  [16]
    0C9C   285 DW750:   $DS_DW750
    0CE0   286 DW780:   $DS_DW780
    0D54   287 DZ11:    $DS_DZ11
    0DBD   288 DZ32:    $DS_DZ32                                                ;  [28]
    0EAE   289 IEU11A:  $DS_IEU11A                                              ;  [39]
    0EF9   290          $DS_KA_DEF
    0EF9   291 KA730:   $DS_KA730                                               ;  [15]
    0FD4   292 KA750:   $DS_KA750
    10A0   293 KA780:   $DS_KA780
    1149   294 KA785:   $DS_KA785                                               ;  [36]
    11F2   295 KAXXX:   $DS_KAXXX                                               ;  [37]
    129B   296 KAZZZ:   $DS_KAZZZ                                               ;  [43]
    1344   297 KMC11:   $DS_KMC11
    1392   298 KW11K:   $DS_KW11K
    13E3   299 LA12:    $DS_LA12                                                ;  [35]
    13FD   300 LA34:    $DS_LA34
    1417   301 LA36:    $DS_LA36
    1431   302 LA38:    $DS_LA38
    144B   303 LA100:   $DS_LA100                                               ;  [35]
    1466   304 LA120:   $DS_LA120
    1481   305 LA180:   $DS_LA180
    149C   306 LESI:    $DS_LESI                                                ;  [34]
    14E8   307 LN01:    $DS_LN01                                                ;  [35]
    1502   308 LP04:    $DS_LP04
    151C   309 LP05:    $DS_LP05
    1536   310 LP06:    $DS_LP06
    1550   311 LP07:    $DS_LP07
```

```
156A    312 LP11:    $DS_LP11
15B7    313 LP14:    $DS_LP14
15D1    314 LP25:    $DS_LP25
15EB    315 LP26:    $DS_LP26
1605    316 LP27:    $DS_LP27
161F    317 LPA11K:  $DS_LPA11K
1678    318 MA780:   $DS_MA780
16F3    319 MBE:     $DS_MBE
1707    320 ML11:    $DS_ML11
1769    321 MS750:   $DS_MS750
177A    322 MS780:   $DS_MS780
17CA    323 PCL11:   $DS_PCL11
1813    324 PX780:   $DS_PX780                                              [19]
187C    325 R80:     $DS_R80                                                [16]
1895    326 RA60:    $DS_RA60
18AF    327 RA80:    $DS_RA80                                               [17]
18C9    328 RA81:    $DS_RA81                                               [21]
18E3    329 RB730:   $DS_RB730                                              [16]
1912    330 ;[46] RC11:      $DS_RC11                                       [38]
1912    331 RCF25:   $DS_RCF25                                              [33]
192D    332 RC25:    $DS_RC25                                               [32]
1947    333 RH750:   $DS_RH750
19A0    334 RH780:   $DS_RH780
1A05    335 RL01:    $DS_RL01
1A1F    336 RL02:    $DS_RL02
1A39    337 RL11:    $DS_RL11
1A86    338 RK06:    $DS_RK06
1AA0    339 RK07:    $DS_RK07
1ABA    340 RK611:   $DS_RK611
1B08    341 RM03:    $DS_RM03
1B2C    342 RM05:    $DS_RM05
1B50    343 RM80:    $DS_RM80
1B74    344 RP04:    $DS_RP04
1B98    345 RP05:    $DS_RP05
1BBC    346 RP06:    $DS_RP06
1BE0    347 RP07:    $DS_RP07
1C04    348 RX02:    $DS_RX02
1C1E    349 RX211:   $DS_RX211
1C6C    350 SBIA:    $DS_SBIA                                               [37]
1CAE    351 TAPE:    $DS_TAPE                                               [42]
1CBE    352 TE16:    $DS_TE16
1CD8    353 TM03:    $DS_TM03
1D06    354 TM78:    $DS_TM78
1D34    355 TS04:    $DS_TS04
1D8B    356 TS05:    $DS_TS05                                               [43]
1DF5    357 TS11:    $DS_TS11
1E4C    358 TU45:    $DS_TU45
1E66    359 TU58:    $DS_TU58
1E80    360 TU77:    $DS_TU77
1E9A    361 TU78:    $DS_TU78
1EB4    362 TU80     $DS_TU80                                               [28]
1F23    363 TU81     $DS_TU81                                               [29]
1F7A    364 UBE:     $DS_UBE
1FB5    365 UDA50:   $DS_UDA50                                              [17]
201E    366 UNA11:   $DS_UNA11                                              [27]
206C    367 VS100:   $DS_VS100                                              [30]
20C4    368 VS125:   $DS_VS125                                              [40]
```

```
211C    369 VS300:  $DS_VS300                       ;                                   [33]
2174    370 VT50:   $DS_VT50
218E    371 VT52:   $DS_VT52
21A8    372 VT55:   $DS_VT55
21C2    373 VT100:  $DS_VT100
21DD    374 VT220:  $DS_VT220                       ;                                   [35]
21F8    375 VT240:  $DS_VT240                       ;                                   [35]
2213    376         .END
```

```
$$N                             = 00000071     D
$$_$                            = 00000005     D
AAT1K                             000001C8 R   D   04
AA11K$B_BR                        00000032     D
AA11K$K_LEN                       00000033     D
AD11K                             00000219 R   D   04
AD11K$B_BR                        00000032     D
AD11K$K_LEN                       00000033     D
AL_DEVTYP                         00000000 R   D   04
BIT...                          = 00000005     D
CI$B_BR                           00000033     D
CI$B_NODE                         00000034     D
CI$B_TR                           00000032     D
CI$K_LEN                          00000041     D
CI$L_FUNC                         00000035     D
CI$L_INDEX                        0000003D     D
CI$L_MAINT_ID                     00000039     D
CI750                             000^026A R   D   04
CI780                             000002E9 R   D   04
CI_NODE                           00000370 R   D   04
CONSOLE                           0000042D R   D   04
CONSOLE$K_LEN                     00000032     D
CR11                              0000043B R   D   04
CR11$B_BR                         00000036     D
CR11$K_LEN                        00000037     D
CR11$L_CSR                        00000032     D
DISK                              00000492 R   D   04
DISK$K_LEN                        00000032     D
DL11                              000004A2 R   D   04
DL11$B_BR                         00000036     D
DL11$K_LEN                        00000037     D
DL11$L_CSR                        00000032     D
DMC11                             000004EF R   D   04
DMC11$B_BR                        00000036     D
DMC11$K_LEN                       00000037     D
DMC11$L_CSR                       00000032     D
DMF32                             0000053D R   D   04
DMF32$B_BR                        00000034     D
DMF32$B_COMMON                    00000032     D
DMF32$B_FLAGS                     00000033     D
DMF32$K_LEN                       00000039     D
DMF32$L_CSR                       00000035     D
DMF32A                            00000590 R   D   04
DMF32A$B_ACTIV                    00000033     D
DMF32A$B_BR                       00000032     D
DMF32A$B_JUMP                     00000037     D
DMF32A$B_SPEED                    00000034     D
DMF32A$K_LEN                      00000038     D
DMF32A$W_FLAGS                    00000035     D
DMF32P                            00000682 R   D   04
DMF32P$B_BR                       00000034     D
DMF32P$B_COMMON                   00000032     D
DMF32P$B_FLAGS                    00000033     D
DMF32P$K_LEN                      00000039     D
DMF32P$L_CSR                      00000035     D
DMF32S                            0000072C R   D   04
DMF32S$B_BR                       00000034     D
DMF32S$B_COMMON                   00000032     D
DMF32S$B_FLAGS                    00000033     D
DMF32S$K_LEN                      00000039     D
DMF32S$L_CSR                      00000035     D
DMP11                             000007A0 R   D   04
DMP11$B_BR                        00000034     D
DMP11$B_COMMON                    00000032     D
DMP11$B_FLAGS                     00000033     D
DMP11$K_LEN                       00000039     D
DMP11$L_CSR                       00000035     D
DMR11                             00000868 R   D   04
DMR11$B_BR                        00000036     D
DMR11$K_LEN                       00000037     D
DMR11$L_CSR                       00000032     D
DMZ32                             000008B6 R   D   04
DR11B                             00000A14 R   D   04
DR11B$B_BR                        00000036     D
DR11B$K_LEN                       00000037     D
DR11B$L_CSR                       00000032     D
DR11K                             00000A62 R   D   04
DR11K$B_BR                        00000032     D
DR11K$K_LEN                       00000033     D
DR11W                             00000AB3 R   D   04
DR11W$B_BR                        00000036     D
DR11W$K_LEN                       00000037     D
DR11W$L_CSR                       00000032     D
DR750                             00000B01 R   D   04
DR750$B_BR                        00000033     D
DR750$B_CONFIG                    00000035     D
DR750$B_SELF                      00000034     D
DR750$B_SLOT                      00000032     D
DR750$B_UUT                       00000036     D
DR750$K_LEN                       00000037     D
DR780                             00000B8C R   D   04
DR780$B_BR                        00000033     D
DR780$B_CONFIG                    00000035     D
DR780$B_SELF                      00000034     D
DR780$B_TR                        00000032     D
DR780$B_UUT                       00000036     D
DR780$K_LEN                       00000037     D
DS$GA_PTABLE                      00000218 RG  D   03
DS$GA_PTDESC                      00000000 RG  D   04
DS$GA_SELECTED                    00000014 RG  D   03
DS$GA_SELECTS                     00000000 RG  D   03
DUP11                             00000C1F R   D   04
DUP11$B_BR                        00000036     D
DUP11$K_LEN                       00000037     D
DUP11$L_CSR                       00000032     D
DW730                             00000C6D R   D   04
DW730$K_LEN                       00000032     D
DW750                             00000C9C R   D   04
DW750$K_LEN                       00000032     D
DW780                             00000CE0 R   D   04
DW780$B_BR                        00000033     D
DW780$B_TR                        00000032     D
DW780$K_LEN                       00000034     D
DZ11                              00000D54 R   D   04
```

DZ11$B_BR	00000036	D		HP$B_UDA50_BURST	00000037	D	
DZ11$B_MTYPE	00000037	D		HP$B_VS100_BR	00000036	D	
DZ11$K_LEN	00000038	D		HP$B_VS125_BR	00000036	D	
DZ11$L_CSR	00000032	D		HP$B_VS300_BR	00000036	D	
DZ32	00000DBD	R	D	04	HP$K_CI_LEN	00000041	D
DZ32$B_ACTIV	00000033	D		HP$K_DISK_LEN	00000032	D	
DZ32$B_BRLVL	00000032	D		HP$K_DMF32A_LEN	00000038	D	
DZ32$B_SPEED	00000034	D		HP$K_DMF32P_LEN	00000039	D	
DZ32$W_FLAGS	00000035	D		HP$K_DMF32S_LEN	00000039	D	
DZ32$_LEN	00000037	D		HP$K_DMF32_LEN	00000039	D	
HP$A_DEPENDENT	00000032	D		HP$K_DMZ32_LEN	0000003A	D	
HP$A_DEVICE	00000018	D		HP$K_IEU11A_LEN	00000033	D	
HP$A_DVA	0000001C	D		HP$K_LA100_LEN	00000032	D	
HP$A_LINK	00000020	D		HP$K_LA12_LEN	00000032	D	
HP$B_CI_BR	0000C033	D		HP$K_LEN	00000037	D	
HP$B_CI_NODE	00000034	D		HP$K_LESI_LEN	0000C037	D	
HP$B_CI_TR	00000032	D		HP$K_RA60_LEN	00000032	D	
HP$B_DMF32A_ACTIV	00000033	D		HP$K_RA80_LEN	00000032	D	
HP$B_DMF32A_BR	00000032	D		HP$K_RA81_LEN	00000032	D	
HP$B_DMF32A_JUMP	00000037	D		HP$K_RB730_LEN	00000037	D	
HP$B_DMF32A_SPEED	00000034	D		HP$K_RC25_LEN	00000032	D	
HP$B_DMF32P_BR	00000034	D		HP$K_RCF25_LEN	00000032	D	
HP$B_DMF32P_COMMON	00000032	D		HP$K_RH750_LEN	00000033	D	
HP$B_DMF32P_FLAGS	00000033	D		HP$K_RH780_LEN	00000034	D	
HP$B_DMF32S_BR	00000034	D		HP$K_RK06_LEN	00000032	D	
HP$B_DMF32S_COMMON	00000032	D		HP$K_RK07_LEN	00000032	D	
HP$B_DMF32S_FLAGS	00000033	D		HP$K_RK61T_LEN	00000037	D	
HP$B_DMF32_BR	00000034	D		HP$K_RL01_LEN	00000032	D	
HP$B_DMF32_COMMON	00000032	D		HP$K_RL02_LEN	00000032	D	
HP$B_DMF32_FLAGS	00000033	D		HP$K_RL11_LEN	00000037	D	
HP$B_DMZ32_BR	00000032	D		HP$K_RM03_LEN	00000032	D	
HP$B_DMZ32_MODEM	00000039	D		HP$K_RM05_LEN	00000032	D	
HP$B_DMZ32_OCTET0	00000033	D		HP$K_RM80_LEN	00000032	D	
HP$B_DMZ32_OCTET1	00000034	D		HP$K_RP04_LEN	00000032	D	
HP$B_DMZ32_OCTET2	00000035	D		HP$K_RP05_LEN	00000032	D	
HP$B_DMZ32_SPEED	00000036	D		HP$K_RP06_LEN	00000032	D	
HP$B_DRIVE	0000000B	D		HP$K_RP07_LEN	00000032	D	
HP$B_DZ32_ACTIV	00000033	D		HP$K_RX02_LEN	00000032	D	
HP$B_DZ32_BRLVL	00000032	D		HP$K_RX21T_LEN	00000037	D	
HP$B_DZ32_SPEED	00000034	D		HP$K_SBIA_LEN	00000032	D	
HP$B_FLAGS	0000000A	D		HP$K_TAPE_LEN	00000032	D	
HP$B_IEU11A_BR	00000032	D		HP$K_TE16_LEN	00000032	D	
HP$B_LESI_BR	00000036	D		HP$K_TM03_LEN	00000033	D	
HP$B_RB730_BR	00000036	D		HP$K_TM78_LEN	00000033	D	
HP$B_RH750_BR	00000032	D		HP$K_TS04_LEN	00000037	D	
HP$B_RH780_BR	00000033	D		HP$K_TS05_LEN	00000033	D	
HP$B_RH780_TR	00000032	D		HP$K_TS11_LEN	00000037	D	
HP$B_RK611_BR	00000036	D		HP$K_TU45_LEN	00000032	D	
HP$B_RL11_BR	00000036	D		HP$K_TU58_LEN	00000032	D	
HP$B_RX21T_BR	00000036	D		HP$K_TU77_LEN	00000032	D	
HP$B_TM03_DRIVE	00000032	D		HP$K_TU78_LEN	00000032	D	
HP$B_TM78_DRIVE	00000032	D		HP$K_TU80_LEN	00000037	D	
HP$B_TS04_BR	00000036	D		HP$K_TU81_LEN	00000037	D	
HP$B_TS11_BR	00000036	D		HP$K_UBE_LEN	00000036	D	
HP$B_TU80_BR	00000036	D		HP$K_UDA50_LEN	00000038	D	
HP$B_TU81_BR	00000036	D		HP$K_VS100_LEN	00000037	D	
HP$B_UDA50_BR	00000036	D		HP$K_VS125_LEN	00000037	D	

```
HP$K_VS300_LEN                  00000037     D
HP$K_VT100_LEN                  00000032     D
HP$K_VT220_LEN                  00000032     D
HP$K_VT240_LEN                  00000032     D
HP$K_VT50_LEN                   00000032     D
HP$K_VT52_LEN                   00000032     D
HP$K_VT55_LEN                   00000032     D
HP$L_CI_FUNC                    00000035     D
HP$L_CI_INDEX                   0000003D     D
HP$L_CI_MAINT_ID                00000039     D
HP$L_DMF32P_CSR                 00000035     D
HP$L_DMF32S_CSR                 00000035     D
HP$L_DMF32_CSR                  00000035     D
HP$L_LESI_IP                    00000032     D
HP$L_RB730_CSR                  0000C032     D
HP$L_RK611_CSR                  00000032     D
HP$L_RL11_CSR                   00000032     D
HP$L_RX211_CSR                  00000032     D
HP$L_TS04_CSR                   00000032     D
HP$L_TS11_CSR                   00000032     D
HP$L_TU80_CSR                   00000032     D
HP$L_TU81_CSR                   00000032     D
HP$L_UBE_CSR                    00000032     D
HP$L_UDA50_UDAIP                00000032     D
HP$L_VS100_CSR                  00000032     D
HP$L_VS125_CSR                  00000032     D
HP$L_VS300_CSR                  00000032     D
HP$M_ALLOC                    = 00000001     D
HP$Q_DEVICE                     00000000     D
HP$T_DEVICE                     0000000C     D
HP$T_TYPE                       00000026     D
HP$W_DMF32A_FLAGS               00000035     D
HP$W_DMZ32_LOOP                 00000037     D
HP$W_DZ32_FLAGS                 00000035     D
HP$W_SIZE                       00000008     D
HP$W_VECTOR                     00000024     D
IEU1TA                          00000EAE R   D   04
IEU11A$B_BR                     00000032     D
IEU11A$K_LEN                    00000033     D
IOC$AL_SYSUCB                   00000000 RG  D   02
IOC$GL_ADPLIST                  0000000C RG  D   02
IOC$GL_DEVLIST                  00000008 RG  D   02
IOC$GL_DPTLIST                  00000010 RG  D   02
KA$B_ACC_TYPE                   00000042     D
KA$B_RESERVED                   00000043     D
KA$B_SID_TYPE                   0000003D     D
KA$K_LEN                        00000046     D
KA$L_FLAGS                      00000032     D
KA$L_SID                        0000003A     D
KA$M_CHAR                     = 00000100     D
KA$M_CRC                      = 00000010     D
KA$M_D_FLOAT                  = 00000002     D
KA$M_EDIT                     = 00000080     D
KA$M_F_FLOAT                  = 00000001     D
KA$M_G_FLOAT                  = 00000004     D
KA$M_H_FLOAT                  = 00000008     D
KA$M_PACKED                   = 00000020     D
KA$M_PACK_MUL                 = 00000040     D
KA$M_TODR                     = 00010000     D
KA$V_G_FLOAT                  = 00000002     D
KA$V_H_FLOAT                  = 00000003     D
KA$V_SB_ERR                   = 00000019     D
KA$V_TODR                     = 00000010     D
KA$V_WCS_LD                   = 00000018     D
KA$W_LATENCY                    0000003E     D
KA$W_MEM_SIZE                   00000044     D
KA$W_PROGRESS                   00000040     D
KA$W_RESERVED                   00000038     D
KA$W_WCS                        00000036     D
KA730                           00000EF9 R   D   04
KA750                           00000FD4 R   D   04
KA780                           000010A0 R   D   04
KA785                           00001149 R   D   04
KAXXX                           000011F2 R   D   04
KAZZZ                           0000129B R   D   04
KMC11                           00001344 R   D   04
KMC11$B_BR                      00000036     D
KMC11$K_LEN                     00000037     D
KMC11$L_CSR                     00000032     D
KW11K                           00001392 R   D   04
KW11K$B_BR                      00000032     D
KW11K$K_LEN                     00000033     D
LA100                           0000144B R   D   04
LA100$K_LEN                     00000032     D
LA12                            000013E3 R   D   04
LA12$K_LEN                      00000032     D
LA120                           00001466 R   D   04
LA120$K_LEN                     00000032     D
LA180                           00001481 R   D   04
LA180$K_LEN                     00000032     D
LA34                            000013FD R   D   04
LA34$K_LEN                      00000032     D
LA36                            00001417 R   D   04
LA36$K_LEN                      00000032     D
LA38                            00001431 R   D   04
LA38$K_LEN                      00000032     D
LESI                            0000149C R   D   04
LESI$B_BR                       00000036     D
LESI$K_LEN                      00000037     D
LESI$L_IP                       00000032     D
LN01                            000014E8 R   D   04
LN01$K_LEN                      00000032     D
LP04                            00001502 R   D   04
LP04$K_LEN                      00000032     D
LP05                            0000151C R   D   04
LP05$K_LEN                      00000032     D
LP06                            00001536 R   D   04
LP06$K_LEN                      00000032     D
LP07                            00001550 R   D   04
LP07$K_LEN                      00000032     D
LP11                            0000156A R   D   04
LP11$B_BR                       00000036     D
LP11$K_LEN                      00000037     D
LP11$L_CSR                      00000032     D
```

```
LP14                            000015B7 R   D  04
LP14$K_LEN                      00000032     D
LP25                            000015D1 R   D  04
LP25$K_LEN                      00000032     D
LP26                            000015EB R   D  04
LP26$K_LEN                      00000032     D
LP27                            00001605 R   D  04
LP27$K_LEN                      00000032     D
LPA11K                          0000161F R   D  04
LPA11K$B_BR                     00000036     D
LPA11K$K_LEN                    00000037     D
LPA11K$L_CSR                    00000032     D
MA780                           00001678 R   D  04
MA780$B_BR                      00000033     D
MA780$B_MPM                     00000034     D
MA780$B_PORT                    00000035     D
MA780$B_TR                      00000032     D
MA780$K_LEN                     00000036     D
MAXPT                       =   00000080     D
MBE                             000016F3 R   D  04
MBE$K_LEN                       00000032     D
ML11                            00001707 R   D  04
ML11$B_ARRAY                    00000032     D
ML11$B_CHIP                     00000033     D
ML11$K_LEN                      00000034     D
MS750                           00001769 R   D  04
MS750$K_LEN                     00000032     D
MS780                           0000177A R   D  04
MS780$B_ARRAYS                  00000033     D
MS780$B_TR                      00000032     D
MS780$K_LEN                     00000034     D
PCL11                           000017CA R   D  04
PCL11$B_BR                      00000036     D
PCL11$K_LEN                     00000037     D
PCL11$L_CSR                     00000032     D
PD$_ADD                     =   0000008A     D
PD$_COMPLEMENT              =   00000089     D
PD$_DECIMAL                 =   00000082     D
PD$_END                     =   00000081     D
PD$_FETCH                   =   00000087     D
PD$_HEXADECIMAL             =   00000084     D
PD$_LITERAL                 =   00000086     D
PD$_LOGICAL                 =   0000008B     D
PD$_NAME                    =   0000008D     D
PD$_OCTAL                   =   00000083     D
PD$_START                   =   00000080     D
PD$_STORE                   =   00000088     D
PD$_STRING                  =   00000085     D
PTD$M_CONTROLLER            =   00000002     D
PTD$M_DEVICE                =   00000003     D
PTD$M_ENDDEVICE             =   0000001B     D
PTD$M_INHERIT               =   00000018     D
PTD$M_INHERIT_CON           =   00000010     D
PTD$M_INHERIT_PRE           =   00000008     D
PTD$M_NAME                  =   00000004     D
PTD$M_UNIT                  =   00000001     D
PX780                           00001813 R   D  04
PX780$B_BR                      00000033     D
PX780$B_NODE                    00000034     D
PX780$B_TR                      00000032     D
PX780$K_LEN                     00000035     D
R80                             0000187C R   D  04
R80$K_LEN                       00000032     D
RA60                            00001895 R   D  04
RA60$K_LEN                      00000032     D
RA80                            000018AF R   D  04
RA80$K_LEN                      00000032     D
RA81                            000018C9 R   D  04
RA81$K_LEN                      00000032     D
RB730                           000018E3 R   D  04
RB730$B_BR                      00000036     D
RB730$K_LEN                     00000037     D
RB730$L_CSR                     00000032     D
RC25                            0000192D R   D  04
RC25$K_LEN                      00000032     D
RCF25                           00001912 R   D  04
RCF25$K_LEN                     00000032     D
RH750                           00001947 R   D  04
RH750$B_BR                      00000032     D
RH750$K_LEN                     00000033     D
RH780                           000019A0 R   D  04
RH780$B_BR                      00000033     D
RH780$B_TR                      00000032     D
RH780$K_LEN                     00000034     D
RK06                            00001A86 R   D  04
RK06$K_LEN                      00000032     D
RK07                            00001AA0 R   D  04
RK07$K_LEN                      00000032     D
RK611                           00001ABA R   D  04
RK611$B_BR                      00000036     D
RK611$K_LEN                     00000037     D
RK611$L_CSR                     00000032     D
RL01                            00001A05 R   D  04
RL01$K_LEN                      00000032     D
RL02                            00001A1F R   D  04
RL02$K_LEN                      00000032     D
RL11                            00001A39 R   D  04
RL11$B_BR                       00000036     D
RL11$K_LEN                      00000037     D
RL11$L_CSR                      00000032     D
RM03                            00001B08 R   D  04
RM03$K_LEN                      00000032     D
RM05                            00001B2C R   D  04
RM05$K_LEN                      00000032     D
RM80                            00001B50 R   D  04
RM80$K_LEN                      00000032     D
RP04                            00001B74 R   D  04
RP04$K_LEN                      00000032     D
RP05                            00001B98 R   D  04
RP05$K_LEN                      00000032     D
RP06                            00001BBC R   D  04
RP06$K_LEN                      00000032     D
RP07                            00001BE0 R   D  04
RP07$K_LEN                      00000032     D
```

```
RX02                    00001C04 R  D  04
RX02$K_LEN              00000032    D
RX211                   00001C1E R  D  04
RX211$B_BR              00000036    D
RX211$K_LEN             00000037    D
RX211$L_CSR             00000032    D
SBIA                    00001C6C R  D  04
SBIA$K_LEN              00000032    D
SIZ...                =  00000001    D
TAPE                    00001CAE R  D  04
TAPE$K_LEN              00000032    D
TE16                    00001CBE R  D  04
TE16$K_LEN              00000032    D
TM03                    00001CD8 R  D  04
TM03$B_DRIVE            00000032    D
TM03$K_LEN              00000033    D
TM78                    00001D06 R  D  04
TM78$B_DRIVE            00000032    D
TM78$K_LEN              00000033    D
TS04                    00001D34 R  D  04
TS04$B_BR               00000036    D
TS04$K_LEN              00000037    D
TS04$L_CSR              00000032    D
TS05                    00001D8B R  D  04
TS05$B_BR               00000032    D
TS05$K_LEN              00000033    D
TS11                    00001DF5 R  D  04
TS11$B_BR               00000036    D
TS11$K_LEN              00000037    D
TS11$L_CSR              00000032    D
TU45                    00001E4C R  D  04
TU45$K_LEN              00000032    D
TU58                    00001E66 R  D  04
TU58$K_LEN              00000032    D
TU77                    00001E80 R  D  04
TU77$K_LEN              00000032    D
TU78                    00001E9A R  D  04
TU78$K_LEN              00000032    D
TU80                    00001EB4 R  D  04
TU80$B_BR               00000036    D
TU80$K_LEN              00000037    D
TU81                    00001F23 R  D  04
TU81$B_BR               00000036    D
TU81$K_LEN              00000037    D
TU81$L_CSR              00000032    D
UBE                     00001F7A R  D  04
UBE$K_LEN               00000036    D
UBE$L_CSR               00000032    D
UDA50                   00001FB5 R  D  04
UDA50$B_BR              00000036    D
UDA50$B_BURST           00000037    D
UDA50$K_LEN             00000038    D
UDA50$L_UDAIP           00000032    D
UNA11                   0000201E R  D  04
UNA11$B_BR              00000032    D
UNA11$K_LEN             00000037    D
UNA11$L_CSR             00000033    D

VS100                   0000206C R  D  04
VS100$B_BR              00000036    D
VS100$K_LEN             00000037    D
VS100$L_CSR             00000032    D
VS125                   000020C4 R  D  04
VS125$B_BR              C0000036    D
VS125$K_LEN             00000037    D
VS125$L_CSR             00000032    D
VS300                   0000211C R  D  04
VS300$B_BR              00000036    D
VS300$K_LEN             00000037    D
VS300$L_CSR             00000032    D
VT100                   000021C2 R  D  04
VT100$K_LEN             00000032    D
VT220                   000021DD R  D  04
VT220$K_LEN             00000032    D
VT240                   000021F8 R  D  04
VT240$K_LEN             00000032    D
VT50                    00002174 R  D  04
VT50$K_LEN              00000032    D
VT52                    0000218E R  D  04
VT52$K_LEN              00000032    D
VT55                    000021A8 R  D  04
VT55$K_LEN              00000032    D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000046 (70.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
WORK	0000041C (1052.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00002213 (8723.)	04 (4.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000071	252 (5)	252 (5)
$$_$	=00000005	264 (6)	264 (6)
AA11K	000001C8-R	259 (6)	252 (5)
AA11K$B_BR	00000032		259 (6)
AA11K$K_LEN	00000033		259 (6)
AD11K	00000219-R	260 (6)	252 (5)
AD11K$B_BR	00000032		260 (6)
AD11K$K_LEN	00000033		260 (6)
AL_DEVTYP	00000000-R	252 (5)	
BIT...	=00000005	375 (6)	259 (6) 260 (6) 262 (6) 263 (6) 264 (6) 266 (6) 267 (6) 268 (6) 269 (6) 270 (6) 271 (6) 272 (6) 273 (6) 274 (6) 275 (6) 276 (6) 277 (6) 278 (6) 279 (6) 280 (6) 281 (6) 282 (6) 283 (6) 284 (6) 285 (6) 286 (6) 287 (6) 288 (6) 289 (6) 291 (6) 292 (6) 293 (6) 294 (6) 295 (6) 296 (6) 297 (6) 298 (6) 299 (6) 300 (6) 301 (6) 302 (6) 303 (6) 304 (6) 305 (6) 306 (6) 307 (6) 308 (6) 309 (6) 310 (6) 311 (6) 312 (6) 313 (6) 314 (6) 315 (6) 316 (6) 317 (6) 318 (6) 319 (6) 320 (6) 321 (6) 322 (6) 323 (6) 324 (6) 325 (6) 326 (6) 327 (6) 328 (6) 329 (6) 331 (6) 332 (6) 333 (6) 334 (6) 335 (6) 336 (6) 337 (6) 338 (6) 339 (6) 340 (6) 341 (6) 342 (6) 343 (6) 344 (6) 345 (6) 346 (6) 347 (6) 348 (6) 349 (6) 350 (6) 351 (6) 352 (6) 353 (6) 354 (6) 355 (6) 356 (6) 357 (6) 358 (6) 359 (6) 360 (6) 361 (6) 362 (6) 363 (6) 364 (6) 365 (6) 366 (6) 367 (6) 368 (6) 369 (6) 370 (6) 371 (6) 372 (6) 373 (6) 374 (6) 375 (6)
CI$B_BR	00000033		262 (6) 263 (6)
CI$B_NODE	00000034		262 (6) 263 (6) 264 (6)
CI$B_TR	00000032		262 (6) 263 (6)
CI$K_LEN	00000041		262 (6) 263 (6) 264 (6)
CI$L_FUNC	00000035		262 (6) 263 (6) 264 (6)
CI$L_INDEX	0000003D		264 (6)
CI$L_MAINT_ID	00000039		262 (6) 263 (6) 264 (6)
CI750	0000026A-R	262 (6)	252 (5)
CI780	000002E9-R	263 (6)	252 (5)
CI_NODE	00000370-R	264 (6)	252 (5)
CONSOLE	0000042D-R	265 (6)	252 (5)
CONSOLE$K_LEN	00000032		266 (6)

```
CR11                            0000043B-R  267     (6)     252     (5)
CR11$B_BR                       00000036                    267     (6)
CR11$K_LEN                      00000037                    267     (6)
CR11$L_CSR                      00000032                    267     (6)
DISK                            00000492-R  268     (6)     252     (5)
DISK$K_LEN                      00000032                    268     (6)
DL11                            000004A2-R  269     (6)     252     (5)
DL11$B_BR                       00000036                    269     (6)
DL11$K_LEN                      00000037                    269     (6)
DL11$L_CSR                      00000032                    269     (6)
DMC11                           000004EF-R  270     (6)     252     (5)
DMC11$B_BR                      00000036                    270     (6)
DMC11$K_LEN                     00000037                    270     (6)
DMC11$L_CSR                     00000032                    270     (6)
DMF32                           0000053D-R  271     (6)     252     (5)
DMF32$B_BR                      00000034                    271     (6)
DMF32$B_COMMON                  00000032                    271     (6)
DMF32$K_LEN                     00000039                    271     (6)     274     (6)
DMF32$L_CSR                     00000035                    271     (6)
DMF32A                          00000590-R  272     (6)     252     (5)
DMF32A$B_ACTIV                  00000033                    272     (6)
DMF32A$B_BR                     00000032                    272     (6)
DMF32A$B_JUMP                   00000037                    272     (6)
DMF32A$B_SPEED                  00000034                    272     (6)
DMF32A$K_LEN                    00000038                    272     (6)
DMF32A$W_FLAGS                  00000035                    272     (6)
DMF32P                          000006B2-R  273     (6)     252     (5)
DMF32P$B_BR                     00000034                    273     (6)
DMF32P$B_COMMON                 00000032                    273     (6)
DMF32P$B_FLAGS                  00000033                    273     (6)
DMF32P$K_LEN                    00000039                    273     (6)
DMF32P$L_CSR                    00000035                    273     (6)
DMF32S                          0000072C-R  274     (6)     252     (5)
DMF32S$B_BR                     00000034                    274     (6)
DMF32S$B_FLAGS                  00000033                    274     (6)
DMF32S$L_CSR                    00000035                    274     (6)
DMP11                           000007A0-R  275     (6)     252     (5)
DMP11$B_BR                      00000034                    275     (6)
DMP11$B_COMMON                  00000032                    275     (6)
DMP11$B_FLAGS                   00000033                    275     (6)
DMP11$K_LEN                     00000039                    275     (6)
DMP11$L_CSR                     00000035                    275     (6)
DMR11                           00000868-R  276     (6)     252     (5)
DMR11$B_BR                      00000036                    276     (6)
DMR11$K_LEN                     00000037                    276     (6)
DMR11$L_CSR                     00000032                    276     (6)
DMZ32                           000008B6-R  277     (6)     252     (5)
DR11B                           00000A14-R  278     (6)     252     (5)
DR11B$B_BR                      00000036                    278     (6)
DR11B$K_LEN                     00000037                    278     (6)
DR11B$L_CSR                     00000032                    278     (6)
DR11K                           00000A62-R  279     (6)     252     (5)
DR11K$B_BR                      00000032                    279     (6)
DR11K$K_LEN                     00000033                    279     (6)
DR11W                           00000AB3-R  280     (6)     252     (5)
DR11W$B_BR                      00000036                    280     (6)
DR11W$K_LEN                     00000037                    280     (6)
```

```
DR11W$L_CSR                     00000032                        280     (6)
DR750                           00000B01-R  281     (6)         252     (5)
DR750$B_BR                      00000033                        281     (6)
DR750$B_CONFIG                  00000035                        281     (6)
DR750$B_SELF                    00000034                        281     (6)
DR750$B_SLOT                    00000032                        281     (6)
DR750$B_UUT                     00000036                        281     (6)
DR750$K_LEN                     00000037                        281     (6)
DR780                           00000B8C-R  282     (6)         252     (5)
DR780$B_BR                      00000033                        282     (6)
DR780$B_CONFIG                  00000035                        282     (6)
DR780$B_SELF                    00000034                        282     (6)
DR780$B_TR                      00000032                        282     (6)
DR780$B_UUT                     00000036                        282     (6)
DR780$K_LEN                     00000037                        282     (6)
DS$GA_PTABLE                    00000218-R  218     (4)
DS$GA_PTDESC                    00000000-R  229     (5)
DS$GA_SELECTED                  00000014-R  205     (4)
DS$GA_SELECTS                   00000000-R  196     (4)
DUP11                           00000C1F-R  283     (6)         252     (5)
DUP11$B_BR                      00000036                        283     (6)
DUP11$K_LEN                     00000037                        283     (6)
DUP11$L_CSR                     00000032                        283     (6)
DW730                           00000C6D-R  284     (6)         252     (5)
DW730$K_LEN                     00000032                        284     (6)
DW750                           00000C9C-R  285     (6)         252     (5)
DW750$K_LEN                     00000032                        285     (6)
DW780                           00000CE0-R  286     (6)         252     (5)
DW780$B_BR                      00000033                        286     (6)
DW780$B_TR                      00000032                        286     (6)
DW780$K_LEN                     00000034                        286     (6)
DZ11                            00000D54-R  287     (6)         252     (5)
DZ11$B_BR                       00000036                        287     (6)
DZ11$B_MTYPE                    00000037                        287     (6)
DZ11$K_LEN                      00000038                        287     (6)
DZ11$L_CSR                      00000032                        287     (6)
DZ32                            00000DBD-R  288     (6)         252     (5)
DZ32$B_ACTIV                    00000033                        288     (6)
DZ32$B_BRLVL                    00000032                        288     (6)
DZ32$B_SPEED                    00000034                        288     (6)
DZ32$W_FLAGS                    00000035                        288     (6)
DZ32$_LEN                       00000037                        288     (6)
HP$A_DEVICE                     00000018                        259     (6)     260     (6)     262     (6)     263     (6)
                                                                264     (6)     267     (6)     269     (6)     270     (6)
                                                                271     (6)     272     (6)     273     (6)     274     (6)
                                                                275     (6)     276     (6)     277     (6)     278     (6)
                                                                279     (6)     280     (6)     281     (6)     282     (6)
                                                                283     (6)     284     (6)     285     (6)     286     (6)
                                                                287     (6)     288     (6)     289     (6)     291     (6)
                                                                292     (6)     297     (6)     298     (6)     306     (6)
                                                                312     (6)     317     (6)     318     (6)     319     (6)
                                                                320     (6)     322     (6)     323     (6)     324     (6)
                                                                329     (6)     333     (6)     334     (6)     337     (6)
                                                                340     (6)     341     (6)     342     (6)     343     (6)
                                                                344     (6)     345     (6)     346     (6)     347     (6)
                                                                349     (6)     350     (6)     353     (6)     354     (6)
                                                                355     (6)     356     (6)     357     (6)     362     (6)
```

		363	(6)	364	(6)	365	(6)	366	(6)
		367	(6)	368	(6)	369	(6)		
HP$A_DVA	0000001C	262	(6)	263	(6)	264	(6)	284	(6)
		285	(6)	286	(6)	333	(6)	334	(6)
		350	(6)						
HP$B_DMZ32_BR	00000032	277	(6)						
HP$B_DMZ32_MODEM	00000039	277	(6)						
HP$B_DMZ32_OCTET0	00000033	277	(6)						
HP$B_DMZ32_OCTET1	00000034	277	(6)						
HP$B_DMZ32_OCTET2	00000035	277	(6)						
HP$B_DMZ32_SPEED	00000036	277	(6)						
HP$B_DRIVE	0000000B	264	(6)	285	(6)	286	(6)	319	(6)
		320	(6)	333	(6)	341	(6)	342	(6)
		343	(6)	344	(6)	345	(6)	346	(6)
		347	(6)	350	(6)	353	(6)	354	(6)
HP$B_FLAGS	0000000A	267	(6)	299	(6)	300	(6)	301	(6)
		302	(6)	303	(6)	304	(6)	305	(6)
		307	(6)	308	(6)	309	(6)	310	(6)
		311	(6)	313	(6)	314	(6)	315	(6)
		316	(6)	317	(6)	320	(6)	325	(6)
		326	(6)	327	(6)	328	(6)	331	(6)
		332	(6)	335	(6)	336	(6)	338	(6)
		339	(6)	341	(6)	342	(6)	343	(6)
		344	(6)	345	(6)	346	(6)	347	(6)
		348	(6)	352	(6)	355	(6)	356	(6)
		357	(6)	358	(6)	359	(6)	360	(6)
		361	(6)	362	(6)	363	(6)	367	(6)
		368	(6)	369	(6)	370	(6)	371	(6)
		372	(6)	373	(6)	374	(6)	375	(6)
HP$B_LESI_BR	00000036	306	(6)						
HP$B_TU80_BR	00000036	362	(6)						
HP$B_UDA50_BR	00000036	365	(6)						
HP$B_UDA50_BURST	00000037	365	(6)						
HP$K_DMZ32_LEN	0000003A	277	(6)						
HP$K_LESI_LEN	00000037	306	(6)						
HP$K_UDA50_LEN	00000038	365	(6)						
HP$L_LESI_IP	00000032	306	(6)						
HP$L_TU80_CSR	00000032	362	(6)						
HP$L_UDA50_UDAIP	00000032	365	(6)						
HP$M_ALLOC	=00000001	267	(6)	299	(6)	300	(6)	301	(6)
		302	(6)	303	(6)	304	(6)	305	(6)
		307	(6)	308	(6)	309	(6)	310	(6)
		311	(6)	313	(6)	314	(6)	315	(6)
		316	(6)	317	(6)	320	(6)	325	(6)
		326	(6)	327	(6)	328	(6)	331	(6)
		332	(6)	335	(6)	336	(6)	338	(6)
		339	(6)	341	(6)	342	(6)	343	(6)
		344	(6)	345	(6)	346	(6)	347	(6)
		348	(6)	352	(6)	355	(6)	356	(6)
		357	(6)	358	(6)	359	(6)	360	(6)
		361	(6)	362	(6)	363	(6)	367	(6)
		368	(6)	369	(6)	370	(6)	371	(6)
		372	(6)	373	(6)	374	(6)	375	(6)
HP$W_DMZ32_LOOP	00000037	277	(6)						
HP$W_VECTOR	00000024	259	(6)	260	(6)	262	(6)	263	(6)
		267	(6)	269	(6)	270	(6)	271	(6)
		272	(6)	273	(6)	274	(6)	275	(6)

```
                                                        276  (6)    277  (6)    278  (6)    279  (6)
                                                        280  (6)    281  (6)    282  (6)    283  (6)
                                                        284  (6)    285  (6)    286  (6)    287  (6)
                                                        288  (6)    289  (6)    297  (6)    298  (6)
                                                        306  (6)    312  (6)    317  (6)    318  (6)
                                                        323  (6)    324  (6)    329  (6)    333  (6)
                                                        334  (6)    337  (6)    340  (6)    349  (6)
                                                        350  (6)    355  (6)    356  (6)    357  (6)
                                                        362  (6)    363  (6)    364  (6)    365  (6)
                                                        366  (6)    367  (6)    368  (6)    369  (6)
IEU11A                  00000EAE-R   289   (6)      252  (5)
IEU11A$B_BR             00000032                      289  (6)
IEU11A$K_LEN            00000033                      289  (6)
IOC$AL_SYSUCB           00000000-R   179   (3)
IOC$GL_ADPLIST          0000000C-R   185   (3)
IOC$GL_DEVLIST          00000008-R   183   (3)
IOC$GL_DPTLIST          00000010-R   187   (3)      188  (3)
KA$B_ACC_TYPE           00000042                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$K_LEN                00000046                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$L_FLAGS              00000032                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_CHAR              =00000100                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_CRC               =00000010                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_D_FLOAT           =00000002                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_EDIT              =00000080                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_F_FLOAT           =00000001                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_G_FLOAT           =00000004                      291  (6)
KA$M_H_FLOAT           =00000008                      291  (6)
KA$M_PACKED            =00000020                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_PACK_MUL          =00000040                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$M_TODR              =00010000                      293  (6)    294  (6)    295  (6)    296  (6)
KA$V_G_FLOAT           =00000002                      292  (6)    293  (6)    294  (6)    295  (6)
                                                      296  (6)
KA$V_H_FLOAT           =00000003                      292  (6)    293  (6)    294  (6)    295  (6)
                                                      296  (6)
KA$V_SB_ERR            =00000019                      291  (6)
KA$V_TODR              =00000010                      291  (6)    292  (6)
KA$V_WCS_LD            =00000018                      291  (6)
KA$W_LATENCY            0000003E                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$W_MEM_SIZE           00000044                      291  (6)
KA$W_PROGRESS           00000040                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA$W_WCS                00000036                      291  (6)    292  (6)    293  (6)    294  (6)
                                                      295  (6)    296  (6)
KA730                   00000EF9-R   291   (6)      252  (5)
KA750                   00000FD4-R   292   (6)      252  (5)
KA780                   000010AU-R   293   (6)      252  (5)
```

KA785	00001149-R	294	(6)	252	(5)
KAXXX	000011F2-R	295	(6)	252	(5)
KAZZZ	0000129B-R	296	(6)	252	(5)
KMC11	00001344-R	297	(6)	252	(5)
KMC11$B_BR	00000036			297	(6)
KMC11$K_LEN	00000037			297	(6)
KMC11$L_CSR	00000032			297	(6)
KW11K	00001392-R	298	(6)	252	(5)
KW11K$B_BR	00000032			298	(6)
KW11K$K_LEN	00000033			298	(6)
LA100	0000144B-R	303	(6)	252	(5)
LA100$K_LEN	00000032			303	(6)
LA12	000013E3-R	299	(6)	252	(5)
LA12$K_LEN	00000032			299	(6)
LA120	00001466-R	304	(6)	252	(5)
LA120$K_LEN	00000032			304	(6)
LA180	00001481-R	305	(6)	252	(5)
LA180$K_LEN	00000032			305	(6)
LA34	000013FD-R	300	(6)	252	(5)
LA34$K_LEN	00000032			300	(6)
LA36	00001417-R	301	(6)	252	(5)
LA36$K_LEN	00000032			301	(6)
LA38	00001431-R	302	(6)	252	(5)
LA38$K_LEN	00000032			302	(6)
LESI	0000149C-R	306	(6)	252	(5)
LN01	000014E8-R	307	(6)	252	(5)
LN01$K_LEN	00000032			307	(6)
LP04	00001502-R	308	(6)	252	(5)
LP04$K_LEN	00000032			308	(6)
LP05	0000151C-R	309	(6)	252	(5)
LP05$K_LEN	00000032			309	(6)
LP06	00001536-R	310	(6)	252	(5)
LP06$K_LEN	00000032			310	(6)
LP07	00001550-R	311	(6)	252	(5)
LP07$K_LEN	00000032			311	(6)
LP11	0000156A-R	312	(6)	252	(5)
LP11$B_BR	00000036			312	(6)
LP11$K_LEN	00000037			312	(6)
LP11$L_CSR	00000032			312	(6)
LP14	000015B7-R	313	(6)	252	(5)
LP14$K_LEN	00000032			313	(6)
LP25	000015D1-R	314	(6)	252	(5)
LP25$K_LEN	00000032			314	(6)
LP26	000015EB-R	315	(6)	252	(5)
LP26$K_LEN	00000032			315	(6)
LP27	00001605-R	316	(6)	252	(5)
LP27$K_LEN	00000032			316	(6)
LPA11K	0000161F-R	317	(6)	252	(5)
LPA11K$B_BR	00000036			317	(6)
LPA11K$K_LEN	00000037			317	(6)
LPA11K$L_CSR	00000032			317	(6)
MA780	00001678-R	318	(6)	252	(5)
MA780$B_BR	00000033			318	(6)
MA780$B_MPM	00000034			318	(6)
MA780$B_PORT	00000035			318	(6)
MA780$B_TR	00000032			318	(6)
MA780$K_LEN	00000036			318	(6)

```
MAXPT                           =00000080      195     (4)      197     (4)      206     (4)      207     (4)      219     (4)
                                                                220     (4)
MBE                              000016F3-R    319     (6)      252     (5)
MBE$K_LEN                        00000032                       319     (6)
ML11                             00001707-R    320     (6)      252     (5)
ML11$B_ARRAY                     00000032                       320     (6)
ML11$B_CHIP                      00000033                       320     (6)
ML11$K_LEN                       00000034                       320     (6)
MS750                            00001769-R    321     (6)      252     (5)
MS750$K_LEN                      00000032                       321     (6)
MS780                            0000177A-R    322     (6)      252     (5)
MS780$B_ARRAYS                   00000033                       322     (6)
MS780$B_TR                       00000032                       322     (6)
MS780$K_LEN                      00000034                       322     (6)
PCL11                            000017CA-R    323     (6)      252     (5)
PCL11$B_BR                       00000036                       323     (6)
PCL11$K_LEN                      00000037                       323     (6)
PCL11$L_CSR                      00000032                       323     (6)
PD$_ADD                         =0000008A      375     (6)      264     (6)      285     (6)      350     (6)
PD$_CASE                        =0000008C      375     (6)      264     (6)
PD$_COMPLEMENT                  =00000089      375     (6)      285     (6)
PD$_DECIMAL                     =00000082      375     (6)      259     (6)      260     (6)      262     (6)      263     (6)
                                                                264     (6)      267     (6)      269     (6)      270     (6)
                                                                271     (6)      272     (6)      273     (6)      274     (6)
                                                                275     (6)      276     (6)      277     (6)      278     (6)
                                                                279     (6)      280     (6)      281     (6)      282     (6)
                                                                283     (6)      286     (6)      287     (6)      289     (6)
                                                                291     (6)      292     (6)      293     (6)      294     (6)
                                                                295     (6)      296     (6)      297     (6)      298     (6)
                                                                306     (6)      312     (6)      317     (6)      318     (6)
                                                                320     (6)      322     (6)      323     (6)      324     (6)
                                                                333     (6)      334     (6)      337     (6)      340     (6)
                                                                349     (6)      353     (6)      354     (6)      355     (6)
                                                                356     (6)      357     (6)      362     (6)      363     (6)
                                                                365     (6)      366     (6)      367     (6)      368     (6)
                                                                369     (6)
PD$_END                         =00000081      375     (6)      259     (6)      260     (6)      262     (6)      263     (6)
                                                                264     (6)      266     (6)      267     (6)      [illegible] (6)
                                                                269     (6)      270     (6)      271     (6)      [illegible]72 (6)
                                                                273     (6)      274     (6)      275     (6)      [illegible]76 (6)
                                                                277     (6)      278     (6)      279     (6)      [illegible]80 (6)
                                                                281     (6)      282     (6)      283     (6)      [illegible]84 (6)
                                                                285     (6)      286     (6)      287     (6)      288     (6)
                                                                289     (6)      291     (6)      292     (6)      293     (6)
                                                                294     (6)      295     (6)      296     (6)      297     (6)
                                                                298     (6)      299     (6)      300     (6)      301     (6)
                                                                302     (6)      303     (6)      304     (6)      305     (6)
                                                                306     (6)      307     (6)      308     (6)      309     (6)
                                                                310     (6)      311     (6)      312     (6)      313     (6)
                                                                314     (6)      315     (6)      316     (6)      317     (6)
                                                                318     (6)      319     (6)      320     (6)      321     (6)
                                                                322     (6)      323     (6)      324     (6)      325     (6)
                                                                326     (6)      327     (6)      328     (6)      329     (6)
                                                                331     (6)      332     (6)      333     (6)      334     (6)
                                                                335     (6)      336     (6)      337     (6)      338     (6)
                                                                339     (6)      340     (6)      341     (6)      342     (6)
                                                                343     (6)      344     (6)      345     (6)      346     (6)
```

```
                                                347     (6)     348     (6)     349     (6)     350     (6)
                                                351     (6)     352     (6)     353     (6)     354     (6)
                                                355     (6)     356     (6)     357     (6)     358     (6)
                                                359     (6)     360     (6)     361     (6)     362     (6)
                                                363     (6)     364     (6)     365     (6)     366     (6)
                                                367     (6)     368     (6)     369     (6)     370     (6)
                                                371     (6)     372     (6)     373     (6)     374     (6)
                                                375     (6)
PD$_FETCH                =00000087      375     (6)     262     (6)     263     (6)     264     (6)     285     (6)
                                                286     (6)     319     (6)     320     (6)     333     (6)
                                                334     (6)     341     (6)     342     (6)     343     (6)
                                                344     (6)     345     (6)     346     (6)     347     (6)
                                                350     (6)
PD$_HEXADECIMAL          =00000084      375     (6)     291     (6)     292     (6)     293     (6)     294     (6)
                                                295     (6)     296     (6)
PD$_LITERAL              =00000086      375     (6)     262     (6)     263     (6)     264     (6)     267     (6)
                                                271     (6)     273     (6)     275     (6)     277     (6)
                                                281     (6)     282     (6)     284     (6)     285     (6)
                                                286     (6)     291     (6)     292     (6)     293     (6)
                                                294     (6)     295     (6)     296     (6)     299     (6)
                                                300     (6)     301     (6)     302     (6)     303     (6)
                                                304     (6)     305     (6)     307     (6)     308     (6)
                                                309     (6)     310     (6)     311     (6)     313     (6)
                                                314     (6)     315     (6)     316     (6)     317     (6)
                                                318     (6)     320     (6)     322     (6)     324     (6)
                                                325     (6)     326     (6)     327     (6)     328     (6)
                                                329     (6)     331     (6)     332     (6)     333     (6)
                                                334     (6)     335     (6)     336     (6)     338     (6)
                                                339     (6)     341     (6)     342     (6)     343     (6)
                                                344     (6)     345     (6)     346     (6)     347     (6)
                                                348     (6)     350     (6)     352     (6)     355     (6)
                                                356     (6)     357     (6)     358     (6)     359     (6)
                                                360     (6)     361     (6)     362     (6)     363     (6)
                                                367     (6)     368     (6)     369     (6)     370     (6)
                                                371     (6)     372     (6)     373     (6)     374     (6)
                                                375     (6)
PD$_LOGICAL              =0000008B      375     (6)     272     (6)     277     (6)     281     (6)     282     (6)
                                                291     (6)     292     (6)     293     (6)     294     (6)
                                                295     (6)     296     (6)
PD$_NAME                 =0000008D      375     (6)     259     (6)     260     (6)     262     (6)     263     (6)
                                                264     (6)     267     (6)     268     (6)     269     (6)
                                                270     (6)     272     (6)     273     (6)     274     (6)
                                                275     (6)     276     (6)     277     (6)     278     (6)
                                                279     (6)     280     (6)     281     (6)     282     (6)
                                                283     (6)     284     (6)     285     (6)     286     (6)
                                                287     (6)     288     (6)     289     (6)     291     (6)
                                                292     (6)     293     (6)     294     (6)     295     (6)
                                                296     (6)     297     (6)     298     (6)     299     (6)
                                                300     (6)     301     (6)     302     (6)     303     (6)
                                                304     (6)     305     (6)     306     (6)     307     (6)
                                                308     (6)     309     (6)     310     (6)     311     (6)
                                                312     (6)     313     (6)     314     (6)     315     (6)
                                                316     (6)     317     (6)     318     (6)     320     (6)
                                                321     (6)     322     (6)     324     (6)     325     (6)
                                                326     (6)     327     (6)     328     (6)     329     (6)
                                                331     (6)     332     (6)     333     (6)     334     (6)
                                                335     (6)     336     (6)     337     (6)     338     (6)
```

```
                                                339     (6)     340     (6)     341     (6)     342     (6)
                                                343     (6)     344     (6)     345     (6)     346     (6)
                                                347     (6)     348     (6)     349     (6)     350     (6)
                                                351     (6)     352     (6)     353     (6)     354     (6)
                                                355     (6)     356     (6)     357     (6)     358     (6)
                                                359     (6)     360     (6)     361     (6)     362     (6)
                                                363     (6)     364     (6)     365     (6)     366     (6)
                                                367     (6)     368     (6)     369     (6)     370     (6)
                                                371     (6)     372     (6)     373     (6)     374     (6)
                                                375     (6)

PD$_OCTAL               =00000083       375     (6)     259     (6)     260     (6)     267     (6)     269     (6)
                                                270     (6)     271     (6)     272     (6)     273     (6)
                                                274     (6)     275     (6)     276     (6)     277     (6)
                                                278     (6)     279     (6)     280     (6)     283     (6)
                                                287     (6)     288     (6)     289     (6)     297     (6)
                                                298     (6)     306     (6)     312     (6)     317     (6)
                                                323     (6)     337     (6)     340     (6)     349     (6)
                                                355     (6)     356     (6)     357     (6)     362     (6)
                                                363     (6)     364     (6)     365     (6)     366     (6)
                                                367     (6)     368     (6)     369     (6)

PD$_START               =00000080       375     (6)     259     (6)     260     (6)     262     (6)     263     (6)
                                                264     (6)     266     (6)     267     (6)     268     (6)
                                                269     (6)     270     (6)     271     (6)     272     (6)
                                                273     (6)     274     (6)     275     (6)     276     (6)
                                                277     (6)     278     (6)     279     (6)     280     (6)
                                                281     (6)     282     (6)     283     (6)     284     (6)
                                                285     (6)     286     (6)     287     (6)     288     (6)
                                                289     (6)     291     (6)     292     (6)     293     (6)
                                                294     (6)     295     (6)     296     (6)     297     (6)
                                                298     (6)     299     (6)     300     (6)     301     (6)
                                                302     (6)     303     (6)     304     (6)     305     (6)
                                                306     (6)     307     (6)     308     (6)     309     (6)
                                                310     (6)     311     (6)     312     (6)     313     (6)
                                                314     (6)     315     (6)     316     (6)     317     (6)
                                                318     (6)     319     (6)     320     (6)     321     (6)
                                                322     (6)     323     (6)     324     (6)     325     (6)
                                                326     (6)     327     (6)     328     (6)     329     (6)
                                                331     (6)     332     (6)     333     (5)     334     (6)
                                                335     (6)     336     (6)     337     (6)     338     (6)
                                                339     (6)     340     (6)     341     (6)     342     (6)
                                                343     (6)     344     (6)     345     (6)     346     (6)
                                                347     (6)     348     (6)     349     (6)     350     (6)
                                                351     (6)     352     (6)     353     (6)     354     (6)
                                                355     (6)     356     (6)     357     (6)     358     (6)
                                                359     (6)     360     (6)     361     (6)     362     (6)
                                                363     (6)     364     (6)     365     (6)     366     (6)
                                                367     (6)     368     (6)     369     (6)     370     (6)
                                                371     (6)     372     (6)     373     (6)     374     (6)
                                                375     (6)

PD$_STORE               =00000088       375     (6)     259     (6)     260     (6)     262     (6)     263     (6)
                                                264     (6)     267     (6)     269     (6)     270     (6)
                                                271     (6)     272     (6)     273     (6)     274     (6)
                                                275     (6)     276     (6)     277     (6)     278     (6)
                                                279     (6)     280     (6)     281     (6)     282     (6)
                                                283     (6)     284     (6)     285     (6)     286     (6)
                                                287     (6)     288     (6)     289     (6)     291     (6)
                                                292     (6)     293     (6)     294     (6)     295     (6)
```

				296 (6) 297 (6) 298 (6) 299 (6) 300 (6) 301 (6) 302 (6) 303 (6) 304 (6) 305 (6) 306 (6) 307 (6) 308 (6) 309 (6) 310 (6) 311 (6) 312 (6) 313 (6) 314 (6) 315 (6) 316 (6) 317 (6) 318 (6) 319 (6) 320 (6) 322 (6) 323 (6) 324 (6) 325 (6) 326 (6) 327 (6) 328 (6) 329 (6) 331 (6) 332 (6) 333 (6) 334 (6) 335 (6) 336 (6) 337 (6) 338 (6) 339 (6) 340 (6) 341 (6) 342 (6) 343 (6) 344 (6) 345 (6) 346 (6) 347 (6) 348 (6) 349 (6) 350 (6) 352 (6) 353 (6) 354 (6) 355 (6) 356 (6) 357 (6) 358 (6) 359 (6) 360 (6) 361 (6) 362 (6) 363 (6) 364 (6) 365 (6) 366 (6) 367 (6) 368 (6) 369 (6) 370 (6) 371 (6) 372 (6) 373 (6) 374 (6) 375 (6)
PD$_STRING	=00000085	375	(6)	264 (6) 272 (6) 273 (6) 274 (6) 275 (6) 277 (6) 281 (6) 282 (6) 287 (6) 320 (6)
PTD$M_CONTROLLER	=00000002	375	(6)	259 (6) 260 (6) 262 (6) 263 (6) 264 (6) 266 (6) 267 (6) 268 (6) 269 (6) 270 (6) 271 (6) 272 (6) 273 (6) 274 (6) 275 (6) 276 (6) 277 (6) 278 (6) 279 (6) 280 (6) 281 (6) 282 (6) 283 (6) 284 (6) 285 (6) 286 (6) 287 (6) 288 (6) 289 (6) 291 (6) 292 (6) 293 (6) 294 (6) 295 (6) 296 (6) 297 (6) 298 (6) 299 (6) 300 (6) 301 (6) 302 (6) 303 (6) 304 (6) 305 (6) 306 (6) 307 (6) 308 (6) 309 (6) 310 (6) 311 (6) 312 (6) 313 (6) 314 (6) 315 (6) 316 (6) 317 (6) 318 (6) 319 (6) 320 (6) 321 (6) 322 (6) 323 (6) 324 (6) 325 (6) 326 (6) 327 (6) 328 (6) 329 (6) 331 (6) 332 (6) 333 (6) 334 (6) 335 (6) 336 (6) 337 (6) 338 (6) 339 (6) 340 (5) 341 (6) 342 (6) 343 (6) 344 (6) 345 (6) 346 (6) 347 (6) 348 (6) 349 (6) 350 (6) 351 (6) 352 (6) 353 (6) 354 (6) 355 (6) 356 (6) 357 (6) 358 (6) 359 (6) 360 (6) 361 (6) 362 (6) 363 (6) 364 (6) 365 (6) 366 (6) 367 (6) 368 (6) 369 (6) 370 (6) 371 (6) 372 (6) 373 (6) 374 (6) 375 (6)
PTD$M_DEVICE	=00000003	375	(6)	259 (6) 260 (6) 262 (6) 263 (6) 267 (6) 268 (6) 270 (6) 274 (6) 275 (6) 276 (6) 278 (6) 279 (6) 280 (6) 281 (6) 282 (6) 283 (6) 297 (6) 298 (6) 317 (6) 320 (6)

				324 (6) 341 (6) 342 (6) 343 (6)
				344 (6) 345 (6) 346 (6) 347 (6)
				351 (6) 355 (6) 356 (6) 357 (6)
				359 (6) 362 (6) 363 (6) 364 (6)
				366 (6)
PTD$M_ENDDEVICE	=0000001B	375	(6)	299 (6) 300 (6) 301 (6) 302 (6)
				303 (6) 304 (6) 305 (6) 307 (6)
				308 (6) 309 (6) 310 (6) 311 (6)
				313 (6) 314 (6) 315 (6) 316 (6)
				325 (6) 327 (6) 328 (6) 331 (6)
				332 (6) 335 (6) 336 (6) 338 (6)
				339 (6) 348 (6) 352 (6) 358 (6)
				360 (6) 361 (6) 367 (6) 368 (6)
				369 (6) 370 (6) 371 (6) 372 (6)
				373 (6) 374 (6) 375 (6)
PTD$M_INHERIT	=00000018	375	(6)	259 (6) 260 (6) 262 (6) 263 (6)
				264 (6) 266 (6) 267 (6) 268 (6)
				269 (6) 270 (6) 271 (6) 272 (6)
				273 (6) 274 (6) 275 (6) 276 (6)
				277 (6) 278 (6) 279 (6) 280 (6)
				281 (6) 282 (6) 283 (6) 284 (6)
				285 (6) 286 (6) 287 (6) 288 (6)
				289 (6) 291 (6) 292 (6) 293 (6)
				294 (6) 295 (6) 296 (6) 297 (6)
				298 (6) 299 (6) 300 (6) 301 (6)
				302 (6) 303 (6) 304 (6) 305 (6)
				306 (6) 307 (6) 308 (6) 309 (6)
				310 (6) 311 (6) 312 (6) 313 (6)
				314 (6) 315 (6) 316 (6) 317 (6)
				318 (6) 319 (6) 320 (6) 321 (6)
				322 (6) 323 (6) 324 (6) 325 (6)
				326 (6) 327 (6) 328 (6) 329 (6)
				331 (6) 332 (6) 333 (6) 334 (6)
				335 (6) 336 (6) 337 (6) 338 (6)
				339 (6) 340 (6) 341 (6) 342 (6)
				343 (6) 344 (6) 345 (6) 346 (6)
				347 (6) 348 (6) 349 (6) 350 (6)
				351 (6) 352 (6) 353 (6) 354 (6)
				355 (6) 356 (6) 357 (6) 358 (6)
				359 (6) 360 (6) 361 (6) 362 (6)
				363 (6) 364 (6) 365 (6) 366 (6)
				367 (6) 368 (6) 369 (6) 370 (6)
				371 (6) 372 (6) 373 (6) 374 (6)
				375 (6)
PTD$M_INHERITED	=00000008	375	(6)	
PTD$M_INHERIT_CON	=00000010	375	(6)	259 (6) 260 (6) 262 (6) 263 (6)
				264 (6) 266 (6) 267 (6) 268 (6)
				269 (6) 270 (6) 271 (6) 272 (6)
				273 (6) 274 (6) 275 (6) 276 (6)
				277 (6) 278 (6) 279 (6) 280 (6)
				281 (6) 282 (6) 283 (6) 284 (6)
				285 (6) 286 (6) 287 (6) 288 (6)
				289 (6) 291 (6) 292 (6) 293 (6)
				294 (6) 295 (6) 296 (6) 297 (6)
				298 (6) 299 (6) 300 (6) 301 (6)
				302 (6) 303 (6) 304 (6) 305 (6)
				306 (6) 307 (6) 308 (6) 309 (6)

```
                                                310  (6)    311  (6)    312  (6)    313  (6)
                                                314  (6)    315  (6)    316  (6)    317  (6)
                                                318  (6)    319  (6)    320  (6)    321  (6)
                                                322  (6)    323  (6)    324  (6)    325  (6)
                                                326  (6)    327  (6)    328  (6)    329  (6)
                                                331  (6)    332  (6)    333  (6)    334  (6)
                                                335  (6)    336  (6)    337  (6)    338  (6)
                                                339  (6)    340  (6)    341  (6)    342  (6)
                                                343  (6)    344  (6)    345  (6)    346  (6)
                                                347  (6)    348  (6)    349  (6)    350  (6)
                                                351  (6)    352  (6)    353  (6)    354  (6)
                                                355  (6)    356  (6)    357  (6)    358  (6)
                                                359  (6)    360  (6)    361  (6)    362  (6)
                                                363  (6)    364  (6)    365  (6)    366  (6)
                                                367  (6)    368  (6)    369  (6)    370  (6)
                                                371  (6)    372  (6)    373  (6)    374  (6)
                                                375  (6)
PTD$M_INHERIT_PRE       =00000008    375  (6)    259  (6)    260  (6)    262  (6)    263  (6)
                                                264  (6)    266  (6)    267  (6)    268  (6)
                                                269  (6)    270  (6)    271  (6)    272  (6)
                                                273  (6)    274  (6)    275  (6)    276  (6)
                                                277  (6)    278  (6)    279  (6)    280  (6)
                                                281  (6)    282  (6)    283  (6)    284  (6)
                                                285  (6)    286  (6)    287  (6)    288  (6)
                                                289  (6)    291  (6)    292  (6)    293  (6)
                                                294  (6)    295  (6)    296  (6)    297  (6)
                                                298  (6)    299  (6)    300  (6)    301  (6)
                                                302  (6)    303  (6)    304  (6)    305  (6)
                                                306  (6)    307  (6)    308  (6)    309  (6)
                                                310  (6)    311  (6)    312  (6)    313  (6)
                                                314  (6)    315  (6)    316  (6)    317  (6)
                                                318  (6)    319  (6)    320  (6)    321  (6)
                                                322  (6)    323  (6)    324  (6)    325  (6)
                                                326  (6)    327  (6)    328  (6)    329  (6)
                                                331  (6)    332  (6)    333  (6)    334  (6)
                                                335  (6)    336  (6)    337  (6)    338  (6)
                                                339  (6)    340  (6)    341  (6)    342  (6)
                                                343  (6)    344  (6)    345  (6)    346  (6)
                                                347  (6)    348  (6)    349  (6)    350  (6)
                                                351  (6)    352  (6)    353  (6)    354  (6)
                                                355  (6)    356  (6)    357  (6)    358  (6)
                                                359  (6)    360  (6)    361  (6)    362  (6)
                                                363  (6)    364  (6)    365  (6)    366  (6)
                                                367  (6)    368  (6)    369  (6)    370  (6)
                                                371  (6)    372  (6)    373  (6)    374  (6)
                                                375  (6)
PTD$M_NAME              =00000004    375  (6)    264  (6)    351  (6)
PTD$M_UNIT              =00000001    375  (6)    259  (6)    260  (6)    262  (6)    263  (6)
                                                264  (6)    266  (6)    267  (6)    268  (6)
                                                269  (6)    270  (6)    271  (6)    272  (6)
                                                273  (6)    274  (6)    275  (6)    276  (6)
                                                277  (6)    278  (6)    279  (6)    280  (6)
                                                281  (6)    282  (6)    283  (6)    284  (6)
                                                285  (6)    286  (6)    287  (6)    288  (6)
                                                289  (6)    291  (6)    292  (6)    293  (6)
                                                294  (6)    295  (6)    296  (6)    297  (6)
                                                298  (6)    299  (6)    300  (6)    301  (6)
```

```
                                                                302     (6)     303     (6)     304     (6)     305     (6)
                                                                306     (6)     307     (6)     308     (6)     309     (6)
                                                                310     (6)     311     (6)     312     (6)     313     (6)
                                                                314     (6)     315     (6)     316     (6)     317     (6)
                                                                318     (6)     319     (6)     320     (6)     321     (6)
                                                                322     (6)     323     (6)     324     (6)     325     (6)
                                                                326     (6)     327     (6)     328     (6)     329     (6)
                                                                331     (6)     332     (6)     333     (6)     334     (6)
                                                                335     (6)     336     (6)     337     (6)     338     (6)
                                                                339     (6)     340     (6)     341     (6)     342     (6)
                                                                343     (6)     344     (6)     345     (6)     346     (6)
                                                                347     (6)     348     (6)     349     (6)     350     (6)
                                                                351     (6)     352     (6)     353     (6)     354     (6)
                                                                355     (6)     356     (6)     357     (6)     358     (6)
                                                                359     (6)     360     (6)     361     (6)     362     (6)
                                                                363     (6)     364     (6)     365     (6)     366     (6)
                                                                367     (6)     368     (6)     369     (6)     370     (6)
                                                                371     (6)     372     (6)     373     (6)     374     (6)
                                                                375     (6)
PTD$V_CONTROLLER                =00000001       375     (6)
PTD$V_INHERIT_CON               =00000004       375     (6)
PTD$V_INHERIT_PRE               =00000003       375     (6)
PTD$V_NAME                      =00000002       375     (6)
PTD$V_UNIT                      =00000000       375     (6)
PX780                            00001813-R     324     (6)     252     (5)
PX780$B_BR                       00000033                       324     (6)
PX780$B_NODE                     00000034                       324     (6)
PX780$B_TR                       00000032                       324     (6)
PX780$K_LEN                      00000035                       324     (6)
R80                              0000187C-R     325     (6)     252     (5)
R80$K_LEN                        00000032                       325     (6)
RA60                             00001895-R     326     (6)     252     (5)
RA60$K_LEN                       00000032                       326     (6)
RA80                             000018AF-R     327     (6)     252     (5)
RA80$K_LEN                       00000032                       327     (6)
RA81                             000018C9-R     328     (6)     252     (5)
RA81$K_LEN                       00000032                       328     (6)
RB730                            000018E3-R     329     (6)     252     (5)
RB730$B_BR                       00000036                       329     (6)
RB730$K_LEN                      00000037                       329     (6)
RC25                             0000192D-R     332     (6)     252     (5)
RC25$K_LEN                       00000032                       332     (6)
RCF25                            00001912-R     331     (6)     252     (5)
RCF25$K_LEN                      00000032                       331     (6)
RH750                            00001947-R     333     (6)     252     (5)
RH750$B_BR                       00000032                       333     (6)
RH750$K_LEN                      00000033                       333     (6)
RH780                            000019A0-R     334     (6)     252     (5)
RH780$B_BR                       00000033                       334     (6)
RH780$B_TR                       00000032                       334     (6)
RH780$K_LEN                      00000034                       334     (6)
RK06                             00001A86-R     338     (6)     252     (5)
RK06$K_LEN                       00000032                       338     (6)
RK07                             00001AA0-R     339     (6)     252     (5)
RK07$K_LEN                       00000032                       339     (6)
RK611                            00001ABA-R     340     (6)     252     (5)
RK611$B_BR                       00000036                       340     (6)
```

```
RK611$K_LEN                     00000037                        340     (6)
RK611$L_CSR                     00000032                        340     (6)
RL01                            00001A05-R      335     (6)     252     (5)
RL01$K_LEN                      00000032                        335     (6)
RL02                            00001A1F-R      336     (6)     252     (5)
RL02$K_LEN                      00000032                        336     (6)
RL11                            00001A39-R      337     (6)     252     (5)
RL11$B_BR                       00000036                        337     (6)
RL11$K_LEN                      00000037                        337     (6)
RL11$L_CSR                      00000032                        337     (6)
RM03                            00001B08-R      341     (6)     252     (5)
RM03$K_LEN                      00000032                        341     (6)
RM05                            00001B2C-R      342     (6)     252     (5)
RM05$K_LEN                      00000032                        342     (6)
RM80                            00001B50-R      343     (6)     252     (5)
RM80$K_LEN                      00000032                        343     (6)
RP04                            00001B74-R      344     (6)     252     (5)
RP04$K_LEN                      00000032                        344     (6)
RP05                            00001B98-R      345     (6)     252     (5)
RP05$K_LEN                      00000032                        345     (6)
RP06                            00001BBC-R      346     (6)     252     (5)
RP06$K_LEN                      00000032                        346     (6)
RP07                            00001BE0-R      347     (6)     252     (5)
RP07$K_LEN                      00000032                        347     (6)
RX02                            00001C04-R      348     (6)     252     (5)
RX02$K_LEN                      00000032                        348     (6)
RX211                           00001C1E-R      349     (6)     252     (5)
RX211$B_BR                      00000036                        349     (6)
RX211$K_LEN                     00000037                        349     (6)
RX211$L_CSR                     00000032                        349     (6)
SBIA                            00001C6C-R      350     (6)     252     (5)
SBIA$K_LEN                      00000032                        350     (6)
SIZ...                          =00000001       375     (6)     259     (6)
                                                                264     (6)
                                                                269     (6)
                                                                273     (6)
                                                                277     (6)
                                                                281     (6)
                                                                285     (6)
                                                                289     (6)
                                                                294     (6)
                                                                298     (6)
                                                                302     (6)
                                                                306     (6)
                                                                310     (6)
                                                                314     (6)
                                                                318     (6)
                                                                322     (6)
                                                                326     (6)
                                                                331     (6)
                                                                335     (6)
                                                                339     (6)
                                                                343     (6)
                                                                347     (6)
                                                                351     (6)
                                                                355     (6)
                                                                359     (6)
                                                                        260     (6)
                                                                        266     (6)
                                                                        270     (6)
                                                                        274     (6)
                                                                        278     (6)
                                                                        282     (6)
                                                                        286     (6)
                                                                        291     (6)
                                                                        295     (6)
                                                                        299     (6)
                                                                        303     (6)
                                                                        307     (6)
                                                                        311     (6)
                                                                        315     (6)
                                                                        319     (6)
                                                                        323     (6)
                                                                        327     (6)
                                                                        332     (6)
                                                                        336     (6)
                                                                        340     (6)
                                                                        344     (6)
                                                                        348     (6)
                                                                        352     (6)
                                                                        356     (6)
                                                                        360     (6)
                                                                                262     (6)
                                                                                267     (6)
                                                                                271     (6)
                                                                                275     (6)
                                                                                279     (6)
                                                                                283     (6)
                                                                                287     (6)
                                                                                292     (6)
                                                                                296     (6)
                                                                                300     (6)
                                                                                304     (6)
                                                                                308     (6)
                                                                                312     (6)
                                                                                316     (6)
                                                                                320     (6)
                                                                                324     (6)
                                                                                328     (6)
                                                                                333     (6)
                                                                                337     (6)
                                                                                341     (6)
                                                                                345     (6)
                                                                                349     (6)
                                                                                353     (6)
                                                                                357     (6)
                                                                                361     (6)
                                                                                        263     (6)
                                                                                        268     (6)
                                                                                        272     (6)
                                                                                        276     (6)
                                                                                        280     (6)
                                                                                        284     (6)
                                                                                        288     (6)
                                                                                        293     (6)
                                                                                        297     (6)
                                                                                        301     (6)
                                                                                        305     (6)
                                                                                        309     (6)
                                                                                        313     (6)
                                                                                        317     (6)
                                                                                        321     (6)
                                                                                        325     (6)
                                                                                        329     (6)
                                                                                        334     (6)
                                                                                        338     (6)
                                                                                        342     (6)
                                                                                        346     (6)
                                                                                        350     (6)
                                                                                        354     (6)
                                                                                        358     (6)
                                                                                        362     (6)
```

				363	(6)	364	(6)	365	(6)	366	(6)
				367	(6)	368	(6)	369	(6)	370	(6)
				371	(6)	372	(6)	373	(6)	374	(6)
				375	(6)						
TAPE	00001CAE-R	351	(6)	252	(5)						
TAPE$K_LEN	00000032			351	(6)						
TE16	00001CBE-R	352	(6)	252	(5)						
TE16$K_LEN	00000032			352	(6)						
TM03	00001CD8-R	353	(6)	252	(5)						
TM03$B_DRIVE	00000032			353	(6)						
TM03$K_LEN	00000033			353	(6)						
TM78	00001D06-R	354	(6)	252	(5)						
TM78$B_DRIVE	00000032			354	(6)						
TM78$K_LEN	00000033			354	(6)						
TS04	00001D34-R	355	(6)	252	(5)						
TS04$B_BR	00000036			355	(6)						
TS04$K_LEN	00000037			355	(6)						
TS04$L_CSR	00000032			355	(6)						
TS05	00001D8B-R	356	(6)	252	(5)						
TS05$B_BR	00000032			356	(6)						
TS05$K_LEN	00000033			356	(6)						
TS11	00001DF5-R	357	(6)	252	(5)						
TS11$B_BR	00000036			357	(6)						
TS11$K_LEN	00000037			357	(6)						
TS11$L_CSR	00000032			357	(6)						
TU45	00001E4C-R	358	(6)	252	(5)						
TU45$K_LEN	00000032			358	(6)						
TU58	00001E66-R	359	(6)	252	(5)						
TU58$K_LEN	00000032			359	(6)						
TU77	00001E80-R	360	(6)	252	(5)						
TU77$K_LEN	00000032			360	(6)						
TU78	00001E9A-R	361	(6)	252	(5)						
TU78$K_LEN	00000032			361	(6)						
TU80	00001EB4-R	362	(6)	252	(5)						
TU80$K_LEN	00000037			362	(6)						
TU81	00001F23-R	363	(6)	252	(5)						
TU81$B_BR	00000036			363	(6)						
TU81$K_LEN	00000037			363	(6)						
TU81$L_CSR	00000032			363	(6)						
UBE	00001F7A-R	364	(6)	252	(5)						
UBE$K_LEN	00000036			364	(6)						
UBE$L_CSR	00000032			364	(6)						
UDA50	00001FB5-R	365	(6)	252	(5)						
UNA11	0000201E-R	366	(6)	252	(5)						
UNA11$B_BR	00000032			366	(6)						
UNA11$K_LEN	00000037			366	(6)						
UNA11$L_CSR	00000033			366	(6)						
VS100	0000206C-R	367	(6)	252	(5)						
VS100$B_BR	00000036			367	(6)						
VS100$K_LEN	00000037			367	(6)						
VS100$L_CSR	00000032			367	(6)						
VS125	000020C4-R	368	(6)	252	(5)						
VS125$B_BR	00000036			368	(6)						
VS125$K_LEN	00000037			368	(6)						
VS125$L_CSR	00000032			368	(6)						
VS300	0000211C-R	369	(6)	252	(5)						
VS300$B_BR	00000036			369	(6)						

```
VS300$K_LEN                     00000037                        369     (6)
VS300$L_CSR                     00000032                        369     (6)
VT100                           000021C2-R      373     (6)     252     (5)
VT100$K_LEN                     00000032                        373     (6)
VT220                           000021DD-R      374     (6)     252     (5)
VT220$K_LEN                     00000032                        374     (6)
VT240                           000021F8-R      375     (6)     252     (5)
VT240$K_LEN                     00000032                        375     (6)
VT50                            00002174-R      370     (6)     252     (5)
VT50$K_LEN                      00000032                        370     (6)
VT52                            0000218E-R      371     (6)     252     (5)
VT52$K_LEN                      00000032                        371     (6)
VT55                            000021A8-R      372     (6)     252     (5)
VT55$K_LEN                      00000032                        372     (6)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	375 (6)	
$DEFINI	1	168 (2)	168 (2) 259 (6) 260 (6) 261 (6) 266 (6) 267 (6) 268 (6) 269 (6) 270 (6) 271 (6) 272 (6) 273 (6) 274 (6) 275 (6) 276 (6) 277 (6) 278 (6) 279 (6) 280 (6) 281 (6) 282 (6) 283 (6) 284 (6) 285 (6) 286 (6) 287 (6) 288 (6) 289 (6) 290 (6) 297 (6) 298 (6) 299 (6) 300 (6) 301 (6) 302 (6) 303 (6) 304 (6) 305 (6) 306 (6) 307 (6) 308 (6) 309 (6) 310 (6) 311 (6) 312 (6) 313 (6) 314 (6) 315 (6) 316 (6) 317 (6) 318 (6) 319 (6) 320 (6) 321 (6) 322 (6) 323 (6) 324 (6) 325 (6) 326 (6) 327 (6) 328 (6) 329 (6) 331 (6) 332 (6) 333 (6) 334 (6) 335 (6) 336 (6) 337 (6) 338 (6) 339 (6) 340 (6) 341 (6) 342 (6) 343 (6) 344 (6) 345 (6) 346 (6) 347 (6) 348 (6) 349 (6) 350 (6) 351 (6) 352 (6) 353 (6) 354 (6) 355 (6) 356 (6) 357 (6) 358 (6) 359 (6) 360 (6) 361 (6) 362 (6) 363 (6) 364 (6) 365 (6) 366 (6) 367 (6) 368 (6) 369 (6) 370 (6) 371 (6) 372 (6) 373 (6) 374 (6) 375 (6)
$DS_$ADD	1	264 (6)	264 (6) 285 (6) 350 (6)
$DS_$CASE	1	264 (6)	264 (6)
$DS_$COMPLEMENT	1	285 (6)	285 (6)
$DS_$DECIMAL	1	259 (6)	259 (6) 260 (6) 262 (6) 263 (6) 264 (6) 267 (6) 269 (6) 270 (6) 271 (6) 272 (6) 273 (6) 274 (6) 275 (6) 276 (6) 277 (6) 278 (6) 279 (6) 280 (6) 281 (6) 282 (6) 283 (6) 286 (6) 287 (6) 289 (6) 291 (6) 292 (6) 293 (6) 294 (6) 295 (6) 296 (6) 297 (6) 298 (6) 306 (6) 312 (6) 317 (6) 318 (6) 320 (6) 322 (6) 323 (6) 324 (6) 333 (6) 334 (6) 337 (6) 340 (6) 349 (6) 353 (6) 354 (6) 355 (6) 356 (6) 357 (6) 362 (6) 363 (6) 365 (6) 366 (6) 367 (6) 368 (6) 369 (6)
$DS_$END	1	259 (6)	259 (6) 260 (6) 262 (6) 263 (6) 264 (6) 266 (6) 267 (6) 268 (6) 269 (6) 270 (6) 271 (6) 272 (6) 273 (6) 274 (6) 275 (6) 276 (6) 277 (6) 278 (6) 279 (6) 280 (6) 281 (6) 282 (6) 283 (6) 284 (6) 285 (6) 286 (6) 287 (6) 288 (6) 289 (6) 291 (6) 292 (6) 293 (6) 294 (6) 295 (6) 296 (6) 297 (6) 298 (6) 299 (6) 300 (6) 301 (6) 302 (6) 303 (6) 304 (6) 305 (6) 306 (6) 307 (6) 308 (6) 309 (6) 310 (6) 311 (6) 312 (6) 313 (6) 314 (6) 315 (6) 316 (6) 317 (6) 318 (6) 319 (6) 320 (6) 321 (6)

```
                                                322     (6)     323     (6)     324     (6)     325     (6)     326     (6)
                                                327     (6)     328     (6)     329     (6)     331     (6)     332     (6)
                                                333     (6)     334     (6)     335     (6)     336     (6)     337     (6)
                                                338     (6)     339     (6)     340     (6)     341     (6)     342     (6)
                                                343     (6)     344     (6)     345     (6)     346     (6)     347     (6)
                                                348     (6)     349     (6)     350     (6)     351     (6)     352     (6)
                                                353     (6)     354     (6)     355     (6)     356     (6)     357     (6)
                                                358     (6)     359     (6)     360     (6)     361     (6)     362     (6)
                                                363     (6)     364     (6)     365     (6)     366     (6)     367     (6)
                                                368     (6)     369     (6)     370     (6)     371     (6)     372     (6)
                                                373     (6)     374     (6)     375     (6)
$DS_$FETCH              1       262     (6)     262     (6)     263     (6)     264     (6)     285     (6)     286     (6)
                                                319     (6)     320     (6)     333     (6)     334     (6)     341     (6)
                                                342     (6)     343     (6)     344     (6)     345     (6)     346     (6)
                                                347     (6)     350     (6)
$)S_$HEX                1       291     (6)     291     (6)     292     (6)     293     (6)     294     (6)     295     (6)
                                                296     (6)
$DS_$INITIALIZE         1       259     (6)     259     (6)     260     (6)     262     (6)     263     (6)     264     (6)
                                                266     (6)     267     (6)     268     (6)     269     (6)     270     (6)
                                                271     (6)     272     (6)     273     (6)     274     (6)     275     (6)
                                                276     (6)     277     (6)     278     (6)     279     (6)     280     (6)
                                                281     (6)     282     (6)     283     (6)     284     (6)     285     (6)
                                                286     (6)     287     (6)     288     (6)     289     (6)     291     (6)
                                                292     (6)     293     (6)     294     (6)     295     (6)     296     (6)
                                                297     (6)     298     (6)     299     (6)     300     (6)     301     (6)
                                                302     (6)     303     (6)     304     (6)     305     (6)     306     (6)
                                                307     (6)     308     (6)     309     (6)     310     (6)     311     (6)
                                                312     (6)     313     (6)     314     (6)     315     (6)     316     (6)
                                                317     (6)     318     (6)     319     (6)     320     (6)     321     (6)
                                                322     (6)     323     (6)     324     (6)     325     (6)     326     (6)
                                                327     (6)     328     (6)     329     (6)     331     (6)     332     (6)
                                                333     (6)     334     (6)     335     (6)     336     (6)     337     (6)
                                                338     (6)     339     (6)     340     (6)     341     (6)     342     (6)
                                                343     (6)     344     (6)     345     (6)     346     (6)     347     (6)
                                                348     (6)     349     (6)     350     (6)     351     (6)     352     (6)
                                                353     (6)     354     (6)     355     (6)     356     (6)     357     (6)
                                                358     (6)     359     (6)     360     (6)     361     (6)     362     (6)
                                                363     (6)     364     (6)     365     (6)     366     (6)     367     (6)
                                                368     (6)     369     (6)     370     (6)     371     (6)     372     (6)
                                                373     (6)     374     (6)     375     (6)
$DS_$LITERAL            1       262     (6)     262     (6)     263     (6)     264     (6)     267     (6)     271     (6)
                                                273     (6)     275     (6)     277     (6)     281     (6)     282     (6)
                                                284     (6)     285     (6)     286     (6)     291     (6)     292     (6)
                                                293     (6)     294     (6)     295     (6)     296     (6)     299     (6)
                                                300     (6)     301     (6)     302     (6)     303     (6)     304     (6)
                                                305     (6)     307     (6)     308     (6)     309     (6)     310     (6)
                                                311     (6)     313     (6)     314     (6)     315     (6)     316     (6)
                                                317     (6)     318     (6)     320     (6)     322     (6)     324     (6)
                                                325     (6)     326     (6)     327     (6)     328     (6)     329     (6)
                                                331     (6)     332     (6)     333     (6)     334     (6)     335     (6)
                                                336     (6)     338     (6)     339     (6)     341     (6)     342     (6)
                                                343     (6)     344     (6)     345     (6)     346     (6)     347     (6)
                                                348     (6)     350     (6)     352     (6)     355     (6)     356     (6)
                                                357     (6)     358     (6)     359     (6)     360     (6)     361     (6)
                                                362     (6)     363     (6)     367     (6)     368     (6)     369     (6)
                                                370     (6)     371     (6)     372     (6)     373     (6)     374     (6)
                                                375     (6)
```

$DS_$LOGICAL 1 272 (6) 272 (6) 277 (6) 281 (6) 282 (6) 291 (6)
292 (6) 293 (6) 294 (6) 295 (6) 296 (6)
$DS_$NAME 1 259 (6) 259 (6) 260 (6) 262 (6) 263 (6) 264 (6)
267 (6) 268 (6) 269 (6) 270 (6) 272 (6)
273 (6) 274 (6) 275 (6) 276 (6) 277 (6)
278 (6) 279 (6) 280 (6) 281 (6) 282 (6)
283 (6) 284 (6) 285 (6) 286 (6) 287 (6)
288 (6) 289 (6) 291 (6) 292 (6) 293 (6)
294 (6) 295 (6) 296 (6) 297 (6) 298 (6)
299 (6) 300 (6) 301 (6) 302 (6) 303 (6)
304 (6) 305 (6) 306 (6) 307 (6) 308 (6)
309 (6) 310 (6) 311 (6) 312 (6) 313 (6)
314 (6) 315 (6) 316 (6) 317 (6) 318 (6)
320 (6) 321 (6) 322 (6) 324 (6) 325 (6)
326 (6) 327 (6) 328 (6) 329 (6) 331 (6)
332 (6) 333 (6) 334 (6) 335 (6) 336 (6)
337 (6) 338 (6) 339 (6) 340 (6) 341 (6)
342 (6) 343 (6) 344 (6) 345 (6) 346 (6)
347 (6) 348 (6) 349 (6) 350 (6) 351 (6)
352 (6) 353 (6) 354 (6) 355 (6) 356 (6)
357 (6) 358 (6) 359 (6) 360 (6) 361 (6)
362 (6) 363 (6) 364 (6) 365 (6) 366 (6)
367 (6) 368 (6) 369 (6) 370 (6) 371 (6)
372 (6) 373 (6) 374 (6) 375 (6)
$DS_$OCTAL 1 259 (6) 259 (6) 260 (6) 267 (6) 269 (6) 270 (6)
271 (6) 272 (6) 273 (6) 274 (6) 275 (6)
276 (6) 277 (6) 278 (6) 279 (6) 280 (6)
283 (6) 287 (6) 288 (6) 289 (6) 297 (6)
298 (6) 306 (6) 312 (6) 317 (6) 323 (6)
337 (6) 340 (6) 349 (6) 355 (6) 356 (6)
357 (6) 362 (6) 363 (6) 364 (6) 365 (6)
366 (6) 367 (6) 368 (6) 369 (6)
$DS_$STORE 1 259 (6) 259 (6) 260 (6) 262 (6) 263 (6) 264 (6)
267 (6) 269 (6) 270 (6) 271 (6) 272 (6)
273 (6) 274 (6) 275 (6) 276 (6) 277 (6)
278 (6) 279 (6) 280 (6) 281 (6) 282 (6)
283 (6) 284 (6) 285 (6) 286 (6) 287 (6)
288 (6) 289 (6) 291 (6) 292 (6) 293 (6)
294 (6) 295 (6) 296 (6) 297 (6) 298 (6)
299 (6) 300 (6) 301 (6) 302 (6) 303 (6)
304 (6) 305 (6) 306 (6) 307 (6) 308 (6)
309 (6) 310 (6) 311 (6) 312 (6) 313 (6)
314 (6) 315 (6) 316 (6) 317 (6) 318 (6)
319 (6) 320 (6) 322 (6) 323 (6) 324 (6)
325 (6) 326 (6) 327 (6) 328 (6) 329 (6)
331 (6) 332 (6) 333 (6) 334 (6) 335 (6)
336 (6) 337 (6) 338 (6) 339 (6) 340 (6)
341 (6) 342 (6) 343 (6) 344 (6) 345 (6)
346 (6) 347 (6) 348 (6) 349 (6) 350 (6)
352 (6) 353 (6) 354 (6) 355 (6) 356 (6)
357 (6) 358 (6) 359 (6) 360 (6) 361 (6)
362 (6) 363 (6) 364 (6) 365 (6) 366 (6)
367 (6) 368 (6) 369 (6) 370 (6) 371 (6)
372 (6) 373 (6) 374 (6) 375 (6)
$DS_$STRING 1 264 (6) 264 (6) 272 (6) 273 (6) 274 (6) 275 (6)
277 (6) 281 (6) 282 (6) 287 (6) 320 (6)
$DS_AA11K 1 259 (6) 259 (6)

$DS_AA11K_DEF	1	259	(6)	259	(6)
$DS_AD11K	1	260	(6)	260	(6)
$DS_AD11K_DEF	1	260	(6)	260	(6)
$DS_CI750	2	262	(6)	262	(6)
$DS_CI780	3	263	(6)	263	(6)
$DS_CI_DEF	2	261	(6)	261	(6)
$DS_CI_NODE	4	264	(6)	264	(6)
$DS_CONSOLE	1	266	(6)	266	(6)
$DS_CONSOLE_DEF	1	266	(6)	266	(6)
$DS_CR11	1	267	(6)	267	(6)
$DS_CR11_DEF	1	267	(6)	267	(6)
$DS_DEVTYP	1	230	(5)	230	(5)
$DS_DISK	1	268	(6)	268	(6)
$DS_DISK_DEF	1	268	(6)	268	(6)
$DS_DL11	1	269	(6)	269	(6)
$DS_DL11_DEF	1	269	(6)	269	(6)
$DS_DMC11	1	270	(6)	270	(6)
$DS_DMC11_DEF	1	270	(6)	270	(6)
$DS_DMF32	1	271	(6)	271	(6)
$DS_DMF32A	2	272	(6)	272	(6)
$DS_DMF32A_DEF	2	272	(6)	272	(6)
$DS_DMF32P	1	273	(6)	273	(6)
$DS_DMF32P_DEF	1	273	(6)	273	(6)
$DS_DMF32S	1	274	(6)	274	(6)
$DS_DMF32S_DEF	2	274	(6)	274	(6)
$DS_DMF32_DEF	1	271	(6)	271	(6)
$DS_DMP11	2	275	(6)	275	(6)
$DS_DMP11_DEF	1	275	(6)	275	(6)
$DS_DMR11	1	276	(6)	276	(6)
$DS_DMR11_DEF	1	276	(6)	276	(6)
$DS_DMZ32	3	277	(6)	277	(6)
$DS_DMZ32_DEF	2	277	(6)	277	(6)
$DS_DR11B	1	278	(6)	278	(6)
$DS_DR11B_DEF	1	278	(6)	278	(6)
$DS_DR11K	1	279	(6)	279	(6)
$DS_DR11K_DEF	1	279	(6)	279	(6)
$DS_DR11W	1	280	(6)	280	(6)
$DS_DR11W_DEF	1	280	(6)	280	(6)
$DS_DR750	2	281	(6)	281	(6)
$DS_DR750_DEF	1	281	(6)	281	(6)
$DS_DR780	2	282	(6)	282	(6)
$DS_DR780_DEF	1	282	(6)	282	(6)
$DS_DUP11	1	283	(6)	283	(6)
$DS_DUP11_DEF	1	283	(6)	283	(6)
$DS_DW730	1	284	(6)	284	(6)
$DS_DW730_DEF	1	284	(6)	284	(6)
$DS_DW750	1	285	(6)	285	(6)
$DS_DW750_DEF	1	285	(6)	285	(6)
$DS_DW780	2	286	(6)	286	(6)
$DS_DW780_DEF	1	286	(6)	286	(6)
$DS_DZ11	1	287	(6)	287	(6)
$DS_DZ11_DEF	1	287	(6)	287	(6)
$DS_DZ32	2	288	(6)	288	(6)
$DS_DZ32_DEF	1	288	(6)	288	(6)
$DS_HPODEF	2	168	(2)	168	(2)
$DS_IEU11A	1	289	(6)	289	(6)
$DS_IEU11A_DEF	1	289	(6)	289	(6)

$DS_KA730	3	291	(6)	291	(6)
$DS_KA750	3	292	(6)	292	(6)
$DS_KA780	2	293	(6)	293	(6)
$DS_KA785	2	294	(6)	294	(6)
$DS_KAXXX	2	295	(6)	295	(6)
$DS_KAZZZ	2	296	(6)	296	(6)
$DS_KA_DEF	3	290	(6)	290	(6)
$DS_KMC11	1	297	(6)	297	(6)
$DS_KMC11_DEF	1	297	(6)	297	(6)
$DS_KW11K	1	298	(6)	298	(6)
$DS_KW11K_DEF	1	298	(6)	298	(6)
$DS_LA100	1	303	(6)	303	(6)
$DS_LA100_DEF	1	303	(6)	303	(6)
$DS_LA12	1	299	(6)	299	(6)
$DS_LA120	1	304	(6)	304	(6)
$DS_LA120_DEF	1	304	(6)	304	(6)
$DS_LA12_DEF	1	299	(6)	299	(6)
$DS_LA180	1	305	(6)	305	(6)
$DS_LA180_DEF	1	305	(6)	305	(6)
$DS_LA34	1	300	(6)	300	(6)
$DS_LA34_DEF	1	300	(6)	300	(6)
$DS_LA36	1	301	(6)	301	(6)
$DS_LA36_DEF	1	301	(6)	301	(6)
$DS_LA38	1	302	(6)	302	(6)
$DS_LA38_DEF	1	302	(6)	302	(6)
$DS_LESI	1	306	(6)	306	(6)
$DS_LESI_DEF	1	306	(6)	306	(6)
$DS_LN01	1	307	(6)	307	(6)
$DS_LN01_DEF	1	307	(6)	307	(6)
$DS_LP04	1	308	(6)	308	(6)
$DS_LP04_DEF	1	308	(6)	308	(6)
$DS_LP05	1	309	(6)	309	(6)
$DS_LP05_DEF	1	309	(6)	309	(6)
$DS_LP06	1	310	(6)	310	(6)
$DS_LP06_DEF	1	310	(6)	310	(6)
$DS_LP07	1	311	(6)	311	(6)
$DS_LP07_DEF	1	311	(6)	311	(6)
$DS_LP11	1	312	(6)	312	(6)
$DS_LP11_DEF	1	312	(6)	312	(6)
$DS_LP14	1	313	(6)	313	(6)
$DS_LP14_DEF	1	313	(6)	313	(6)
$DS_LP25	1	314	(6)	314	(6)
$DS_LP25_DEF	1	314	(6)	314	(6)
$DS_LP26	1	315	(6)	315	(6)
$DS_LP26_DEF	1	315	(6)	315	(6)
$DS_LP27	1	316	(6)	316	(6)
$DS_LP27_DEF	1	316	(6)	316	(6)
$DS_LPA11K	1	317	(6)	317	(6)
$DS_LPA11K_DEF	1	317	(6)	317	(6)
$DS_MA780	2	318	(6)	318	(6)
$DS_MA780_DEF	1	318	(6)	318	(6)
$DS_MBE	1	319	(6)	319	(6)
$DS_MBE_DEF	1	319	(6)	319	(6)
$DS_ML11	1	320	(6)	320	(6)
$DS_ML11_DEF	1	320	(6)	320	(6)
$DS_MS750	1	321	(6)	321	(6)
$DS_MS750_DEF	1	321	(6)	321	(6)

```
$DS_MS780               1       322     (6)     322     (6)
$DS_MS780_DEF           1       322     (6)     322     (6)
$DS_PCL11               1       323     (6)     323     (6)
$DS_PCL11_DEF           1       323     (6)     323     (6)
$DS_PDDEF               1       259     (6)     259     (6)   260     (6)   262     (6)   263     (6)   264     (6)
                                                266     (6)   267     (6)   268     (6)   269     (6)   270     (6)
                                                271     (6)   272     (6)   273     (6)   274     (6)   275     (6)
                                                276     (6)   277     (6)   278     (6)   279     (6)   280     (6)
                                                281     (6)   282     (6)   283     (6)   284     (6)   285     (6)
                                                286     (6)   287     (6)   288     (6)   289     (6)   291     (6)
                                                292     (6)   293     (6)   294     (6)   295     (6)   296     (6)
                                                297     (6)   298     (6)   299     (6)   300     (6)   301     (6)
                                                302     (6)   303     (6)   304     (6)   305     (6)   306     (6)
                                                307     (6)   308     (6)   309     (6)   310     (6)   311     (6)
                                                312     (6)   313     (6)   314     (6)   315     (6)   316     (6)
                                                317     (6)   318     (6)   319     (6)   320     (6)   321     (6)
                                                322     (6)   323     (6)   324     (6)   325     (6)   326     (6)
                                                327     (6)   328     (6)   329     (6)   331     (6)   332     (6)
                                                333     (6)   334     (6)   335     (6)   336     (6)   337     (6)
                                                338     (6)   339     (6)   340     (6)   341     (6)   342     (6)
                                                343     (6)   344     (6)   345     (6)   346     (6)   347     (6)
                                                348     (6)   349     (6)   350     (6)   351     (6)   352     (6)
                                                353     (6)   354     (6)   355     (6)   356     (6)   357     (6)
                                                358     (6)   359     (6)   360     (6)   361     (6)   362     (6)
                                                363     (6)   364     (6)   365     (6)   366     (6)   367     (6)
                                                368     (6)   369     (6)   370     (6)   371     (6)   372     (6)
                                                373     (6)   374     (6)   375     (6)
$DS_PTDDEF              1       259     (6)     259     (6)   260     (6)   262     (6)   263     (6)   264     (6)
                                                266     (6)   267     (6)   268     (6)   269     (6)   270     (6)
                                                271     (6)   272     (6)   273     (6)   274     (6)   275     (6)
                                                276     (6)   277     (6)   278     (6)   279     (6)   280     (6)
                                                281     (6)   282     (6)   283     (6)   284     (6)   285     (6)
                                                286     (6)   287     (6)   288     (6)   289     (6)   291     (6)
                                                292     (6)   293     (6)   294     (6)   295     (6)   296     (6)
                                                297     (6)   298     (6)   299     (6)   300     (6)   301     (6)
                                                302     (6)   303     (6)   304     (6)   305     (6)   306     (6)
                                                307     (6)   308     (6)   309     (6)   310     (6)   311     (6)
                                                312     (6)   313     (6)   314     (6)   315     (6)   316     (6)
                                                317     (6)   318     (6)   319     (6)   320     (6)   321     (6)
                                                322     (6)   323     (6)   324     (6)   325     (6)   326     (6)
                                                327     (6)   328     (6)   329     (6)   331     (6)   332     (6)
                                                333     (6)   334     (6)   335     (6)   336     (6)   337     (6)
                                                338     (6)   339     (6)   340     (6)   341     (6)   342     (6)
                                                343     (6)   344     (6)   345     (6)   346     (6)   347     (6)
                                                348     (6)   349     (6)   350     (6)   351     (6)   352     (6)
                                                353     (6)   354     (6)   355     (6)   356     (6)   357     (6)
                                                358     (6)   359     (6)   360     (6)   361     (6)   362     (6)
                                                363     (6)   364     (6)   365     (6)   366     (6)   367     (6)
                                                368     (6)   369     (6)   370     (6)   371     (6)   372     (6)
                                                373     (6)   374     (6)   375     (6)
$DS_PX780               2       324     (6)     324     (6)
$DS_PX780_DEF           1       324     (6)     324     (6)
$DS_R80                 1       325     (6)     325     (6)
$DS_R80_DEF             1       325     (6)     325     (6)
$DS_RA60                1       326     (6)     326     (6)
$DS_RA60_DEF            1       326     (6)     326     (6)
$DS_RA80                1       327     (6)     327     (6)
```

$DS_RA80_DEF	1	327	(6)	327	(6)
$DS_RA81	1	328	(6)	328	(6)
$DS_RA81_DEF	1	328	(6)	328	(6)
$DS_RB730	2	329	(6)	329	(6)
$DS_RB730_DEF	1	329	(6)	329	(6)
$DS_RC25	1	332	(6)	332	(6)
$DS_RC25_DEF	1	332	(6)	332	(6)
$DS_RCF25	1	331	(6)	331	(6)
$DS_RCF25_DEF	1	331	(6)	331	(6)
$DS_RH750	2	333	(6)	333	(6)
$DS_RH750_DEF	1	333	(6)	333	(6)
$DS_RH780	2	334	(6)	334	(6)
$DS_RH780_DEF	1	334	(6)	334	(6)
$DS_RK06	1	338	(6)	338	(6)
$DS_RK06_DEF	1	338	(6)	338	(6)
$DS_RK07	1	339	(6)	339	(6)
$DS_RK07_DEF	1	339	(6)	339	(6)
$DS_RK611	1	340	(6)	340	(6)
$DS_RK611_DEF	1	340	(6)	340	(6)
$DS_RL01	1	335	(6)	335	(6)
$DS_RL01_DEF	1	335	(6)	335	(6)
$DS_RL02	1	336	(6)	336	(6)
$DS_RL02_DEF	1	336	(6)	336	(6)
$DS_RL11	1	337	(6)	337	(6)
$DS_RL11_DEF	1	337	(6)	337	(6)
$DS_RM03	1	341	(6)	341	(6)
$DS_RM03_DEF	1	341	(6)	341	(6)
$DS_RM05	1	342	(6)	342	(6)
$DS_RM05_DEF	1	342	(6)	342	(6)
$DS_RM80	1	343	(6)	343	(6)
$DS_RM80_DEF	1	343	(6)	343	(6)
$DS_RP04	1	344	(6)	344	(6)
$DS_RP04_DEF	1	344	(6)	344	(6)
$DS_RP05	1	345	(6)	345	(6)
$DS_RP05_DEF	1	345	(6)	345	(6)
$DS_RP06	1	346	(6)	346	(6)
$DS_RP06_DEF	1	346	(6)	346	(6)
$DS_RP07	1	347	(6)	347	(6)
$DS_RP07_DEF	1	347	(6)	347	(6)
$DS_RX02	1	348	(6)	348	(6)
$DS_RX02_DEF	1	348	(6)	348	(6)
$DS_RX211	1	349	(6)	349	(6)
$DS_RX211_DEF	1	349	(6)	349	(6)
$DS_SBIA	2	350	(6)	350	(6)
$DS_SBIA_DEF	1	350	(6)	350	(6)
$DS_TAPE	1	351	(6)	351	(6)
$DS_TAPE_DEF	1	351	(6)	351	(6)
$DS_TE16	1	352	(6)	352	(6)
$DS_TE16_DEF	1	352	(6)	352	(6)
$DS_TM03	1	353	(6)	353	(6)
$DS_TM03_DEF	1	353	(6)	353	(6)
$DS_TM78	1	354	(6)	354	(6)
$DS_TM78_DEF	1	354	(6)	354	(6)
$DS_TS04	1	355	(6)	355	(6)
$DS_TS04_DEF	1	355	(6)	355	(6)
$DS_TS05	2	356	(6)	356	(6)
$DS_TS05_DEF	1	356	(6)	356	(6)

$DS_TS11	1	357	(6)	357	(6)										
$DS_TS11_DEF	1	357	(6)	357	(6)										
$DS_TU45	1	358	(6)	358	(6)										
$DS_TU45_DEF	1	358	(6)	358	(6)										
$DS_TU58	1	359	(6)	359	(6)										
$DS_TU58_DEF	1	359	(6)	359	(6)										
$DS_TU77	1	360	(6)	360	(6)										
$DS_TU77_DEF	1	360	(6)	360	(6)										
$DS_TU78	1	361	(6)	361	(6)										
$DS_TU78_DEF	1	361	(6)	361	(6)										
$DS_TU80	2	362	(6)	362	(6)										
$DS_TU80_DEF	1	362	(6)	362	(6)										
$DS_TU81	2	363	(6)	363	(6)										
$DS_TU81_DEF	1	363	(6)	363	(6)										
$DS_UBE	1	364	(6)	364	(6)										
$DS_UBE_DEF	1	364	(6)	364	(6)										
$DS_UDA50	1	365	(6)	365	(6)										
$DS_UDA50_DEF	1	365	(6)	365	(6)										
$DS_UNA11	2	366	(6)	366	(6)										
$DS_UNA11_DEF	1	366	(6)	366	(6)										
$DS_VS100	1	367	(6)	367	(6)										
$DS_VS100_DEF	1	367	(6)	367	(6)										
$DS_VS125	1	368	(6)	368	(6)										
$DS_VS125_DEF	1	368	(6)	368	(6)										
$DS_VS300	1	369	(6)	369	(6)										
$DS_VS300_DEF	1	369	(6)	369	(6)										
$DS_VT100	1	373	(6)	373	(6)										
$DS_VT100_DEF	1	373	(6)	373	(6)										
$DS_VT220	1	374	(6)	374	(6)										
$DS_VT220_DEF	1	374	(6)	374	(6)										
$DS_VT240	1	375	(6)	375	(6)										
$DS_VT240_DEF	1	375	(6)	375	(6)										
$DS_VT50	1	370	(6)	370	(6)										
$DS_VT50_DEF	1	370	(6)	370	(6)										
$DS_VT52	1	371	(6)	371	(6)										
$DS_VT52_DEF	1	371	(6)	371	(6)										
$DS_VT55	1	372	(6)	372	(6)										
$DS_VT55_DEF	1	372	(6)	372	(6)										
$EQU	1	375	(6)	259	(6)	260	(6)	262	(6)	263	(6)	264	(6)		
				266	(6)	267	(6)	268	(6)	269	(6)	270	(6)		
				271	(6)	272	(6)	273	(6)	274	(6)	275	(6)		
				276	(6)	277	(6)	278	(6)	279	(6)	280	(6)		
				281	(6)	282	(6)	283	(6)	284	(6)	285	(6)		
				286	(6)	287	(6)	288	(6)	289	(6)	291	(6)		
				292	(6)	293	(6)	294	(6)	295	(6)	296	(6)		
				297	(6)	298	(6)	299	(6)	300	(6)	301	(6)		
				302	(6)	303	(6)	304	(6)	305	(6)	306	(6)		
				307	(6)	308	(6)	309	(6)	310	(6)	311	(6)		
				312	(6)	313	(6)	314	(6)	315	(6)	316	(6)		
				317	(6)	318	(6)	319	(6)	320	(6)	321	(6)		
				322	(6)	323	(6)	324	(6)	325	(6)	326	(6)		
				327	(6)	328	(6)	329	(6)	331	(6)	332	(6)		
				333	(6)	334	(6)	335	(6)	336	(6)	337	(6)		
				338	(6)	339	(6)	340	(6)	341	(6)	342	(6)		
				343	(6)	344	(6)	345	(6)	346	(6)	347	(6)		
				348	(6)	349	(6)	350	(6)	351	(6)	352	(6)		
				353	(6)	354	(6)	355	(6)	356	(6)	357	(6)		

				358	(6)	359	(6)	360	(6)	361	(6)	362	(6)		
				363	(6)	364	(6)	365	(6)	366	(6)	367	(6)		
				368	(6)	369	(6)	370	(6)	371	(6)	372	(6)		
				373	(6)	374	(6)	375	(6)						
$EQULS1	1	375	(6)	259	(6)	260	(6)	262	(6)	263	(6)	264	(6)		
				266	(6)	267	(6)	268	(6)	269	(6)	270	(6)		
				271	(6)	272	(6)	273	(6)	274	(6)	275	(6)		
				276	(6)	277	(6)	278	(6)	279	(6)	280	(6)		
				281	(6)	282	(6)	283	(6)	284	(6)	285	(6)		
				286	(6)	287	(6)	288	(6)	289	(6)	291	(6)		
				292	(6)	293	(6)	294	(6)	295	(6)	296	(6)		
				297	(6)	298	(6)	299	(6)	300	(6)	301	(6)		
				302	(6)	303	(6)	304	(6)	305	(6)	306	(6)		
				307	(6)	308	(6)	309	(6)	310	(6)	311	(6)		
				312	(6)	313	(6)	314	(6)	315	(6)	316	(6)		
				317	(6)	318	(6)	319	(6)	320	(6)	321	(6)		
				322	(6)	323	(6)	324	(6)	325	(6)	326	(6)		
				327	(6)	328	(6)	329	(6)	331	(6)	332	(6)		
				333	(6)	334	(6)	335	(6)	336	(6)	337	(6)		
				338	(6)	339	(6)	340	(6)	341	(6)	342	(6)		
				343	(6)	344	(6)	345	(6)	346	(6)	347	(6)		
				348	(6)	349	(6)	350	(6)	351	(6)	352	(6)		
				353	(6)	354	(6)	355	(6)	356	(6)	357	(6)		
				358	(6)	359	(6)	360	(6)	361	(6)	362	(6)		
				363	(6)	364	(6)	365	(6)	366	(6)	367	(6)		
				368	(6)	369	(6)	370	(6)	371	(6)	372	(6)		
				373	(6)	374	(6)	375	(6)						
$EQULST	1	259	(6)	259	(6)	260	(6)	262	(6)	263	(6)	264	(6)		
				266	(6)	267	(6)	268	(6)	269	(6)	270	(6)		
				271	(6)	272	(6)	273	(6)	274	(6)	275	(6)		
				276	(6)	277	(6)	278	(6)	279	(6)	280	(6)		
				281	(6)	282	(6)	283	(6)	284	(6)	285	(6)		
				286	(6)	287	(6)	288	(6)	289	(6)	291	(6)		
				292	(6)	293	(6)	294	(6)	295	(6)	296	(6)		
				297	(6)	298	(6)	299	(6)	300	(6)	301	(6)		
				302	(6)	303	(6)	304	(6)	305	(6)	306	(6)		
				307	(6)	308	(6)	309	(6)	310	(6)	311	(6)		
				312	(6)	313	(6)	314	(6)	315	(6)	316	(6)		
				317	(6)	318	(6)	319	(6)	320	(6)	321	(6)		
				322	(6)	323	(6)	324	(6)	325	(6)	326	(6)		
				327	(6)	328	(6)	329	(6)	331	(6)	332	(6)		
				333	(6)	334	(6)	335	(6)	336	(6)	337	(6)		
				338	(6)	339	(6)	340	(6)	341	(6)	342	(6)		
				343	(6)	344	(6)	345	(6)	346	(6)	347	(6)		
				348	(6)	349	(6)	350	(6)	351	(6)	352	(6)		
				353	(6)	354	(6)	355	(6)	356	(6)	357	(6)		
				358	(6)	359	(6)	360	(6)	361	(6)	362	(6)		
				363	(6)	364	(6)	365	(6)	366	(6)	367	(6)		
				368	(6)	369	(6)	370	(6)	371	(6)	372	(6)		
				373	(6)	374	(6)	375	(6)						
$GBLINI	2			259	(6)	260	(6)	262	(6)	263	(6)	264	(6)		
				266	(6)	267	(6)	268	(6)	269	(6)	270	(6)		
				271	(6)	272	(6)	273	(6)	274	(6)	275	(6)		
				276	(6)	277	(6)	278	(6)	279	(6)	280	(6)		
				281	(6)	282	(6)	283	(6)	284	(6)	285	(6)		
				286	(6)	287	(6)	288	(6)	289	(6)	291	(6)		
				292	(6)	293	(6)	294	(6)	295	(6)	296	(6)		

		297 (6)	298 (6)	299 (6) 300 (6) 301 (6)
		302 (6)	303 (6)	304 (6) 305 (6) 306 (6)
		307 (6)	308 (6)	309 (6) 310 (6) 311 (6)
		312 (6)	313 (6)	314 (6) 315 (6) 316 (6)
		317 (6)	318 (6)	319 (6) 320 (6) 321 (6)
		322 (6)	323 (6)	324 (6) 325 (6) 326 (6)
		327 (6)	328 (6)	329 (6) 331 (6) 332 (6)
		333 (6)	334 (6)	335 (6) 336 (6) 337 (6)
		338 (6)	339 (6)	340 (6) 341 (6) 342 (6)
		343 (6)	344 (6)	345 (6) 346 (6) 347 (6)
		348 (6)	349 (6)	350 (6) 351 (6) 352 (6)
		353 (6)	354 (6)	355 (6) 356 (6) 357 (6)
		358 (6)	359 (6)	360 (6) 361 (6) 362 (6)
		363 (6)	364 (6)	365 (6) 366 (6) 367 (6)
		368 (6)	369 (6)	370 (6) 371 (6) 372 (6)
		373 (6)	374 (6)	375 (6)
$VIELD	1	259 (6)	260 (6)	262 (6) 263 (6) 264 (6)
		266 (6)	267 (6)	268 (6) 269 (6) 270 (6)
		271 (6)	272 (6)	273 (6) 274 (6) 275 (6)
		276 (6)	277 (6)	278 (6) 279 (6) 280 (6)
		281 (6)	282 (6)	283 (6) 284 (6) 285 (6)
		286 (6)	287 (6)	288 (6) 289 (6) 291 (6)
		292 (6)	293 (6)	294 (6) 295 (6) 296 (6)
		297 (6)	298 (6)	299 (6) 300 (6) 301 (6)
		302 (6)	303 (6)	304 (6) 305 (6) 306 (6)
		307 (6)	308 (6)	309 (6) 310 (6) 311 (6)
		312 (6)	313 (6)	314 (6) 315 (6) 316 (6)
		317 (6)	318 (6)	319 (6) 320 (6) 321 (6)
		322 (6)	323 (6)	324 (6) 325 (6) 326 (6)
		327 (6)	328 (6)	329 (6) 331 (6) 332 (6)
		333 (6)	334 (6)	335 (6) 336 (6) 337 (6)
		338 (6)	339 (6)	340 (6) 341 (6) 342 (6)
		343 (6)	344 (6)	345 (6) 346 (6) 347 (6)
		348 (6)	349 (6)	350 (6) 351 (6) 352 (6)
		353 (6)	354 (6)	355 (6) 356 (6) 357 (6)
		358 (6)	359 (6)	360 (6) 361 (6) 362 (6)
		363 (6)	364 (6)	365 (6) 366 (6) 367 (6)
		368 (6)	369 (6)	370 (6) 371 (6) 372 (6)
		373 (6)	374 (6)	375 (6)
$VIELD1	1	375 (6)	259 (6)	260 (6) 262 (6) 263 (6) 264 (6)
			266 (6)	267 (6) 268 (6) 269 (6) 270 (6)
			271 (6)	272 (6) 273 (6) 274 (6) 275 (6)
			276 (6)	277 (6) 278 (6) 279 (6) 280 (6)
			281 (6)	282 (6) 283 (6) 284 (6) 285 (6)
			286 (6)	287 (6) 288 (6) 289 (6) 291 (6)
			292 (6)	293 (6) 294 (6) 295 (6) 296 (6)
			297 (6)	298 (6) 299 (6) 300 (6) 301 (6)
			302 (6)	303 (6) 304 (6) 305 (6) 306 (6)
			307 (6)	308 (6) 309 (6) 310 (6) 311 (6)
			312 (6)	313 (6) 314 (6) 315 (6) 316 (6)
			317 (6)	318 (6) 319 (6) 320 (6) 321 (6)
			322 (6)	323 (6) 324 (6) 325 (6) 326 (6)
			327 (6)	328 (6) 329 (6) 331 (6) 332 (6)
			333 (6)	334 (6) 335 (6) 336 (6) 337 (6)
			338 (6)	339 (6) 340 (6) 341 (6) 342 (6)
			343 (6)	344 (6) 345 (6) 346 (6) 347 (6)
			348 (6)	349 (6) 350 (6) 351 (6) 352 (6)

353 (6) 354 (6) 355 (6) 356 (6) 357 (6)
358 (6) 359 (6) 360 (6) 361 (6) 362 (6)
363 (6) 364 (6) 365 (6) 366 (6) 367 (6)
368 (6) 369 (6) 370 (6) 371 (6) 372 (6)
373 (6) 374 (6) 375 (6)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.10	00:00:00.28
Command processing	142	00:00:00.79	00:00:01.70
Pass 1	5363	00:02:21.40	00:03:13.74
Symbol table sort	7	00:00:00.81	00:00:01.05
Pass 2	1461	00:00:15.51	00:00:19.77
Symbol table output	75	00:00:00.43	00:00:00.47
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	799	00:00:07.83	00:00:10.19
Assembler run totals	7916	00:02:46.91	00:03:47.24

The working set limit was 1000 pages.
631524 bytes (1234 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 578 non-local and 0 local symbols.
376 source lines were read in Pass 1, producing 0 object records in Pass 2.
1501 pages of virtual memory were used to define 352 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	237
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	242

2534 GETS were required to define 242 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) IOBASE/UPDA=(IOBASE.UPD,IOBASE.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 41 HISTORY ; DETAILED
(2) 87 DECLARATIONS
(3) 123 I/O COMPLETION POSTING
(4) 292.2 VIRTUAL I/O COMPLETION
(5) 301.60 QUEUE NEXT SEGMENT
(6) 639.15 BUFFERED READ COMPLETION AST ROUTINE
(7) 701.75 DIRECT I/O COMPLETION AST ROUTINE
(8) 831.57 MOVE DATA TO USER BUFFER
(9) 831.80 UNLOCK AREAS IN IRPE'S .

```
                        0000     .1          .TITLE  IOPOST  *** IOPOST I/O completion posting
                        0000     .2          .IDENT  /05-04/
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1978, 1979, 1980                                           *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25
                        0000     26 ;++
                        0000     27 ; FACILITY: EXECUTIVE, I/O SYSTEM
                        0000     28 ;
                        0000     29 ; ABSTRACT:
                        0000     30 ;       IOCIOPOST IMPLEMENTS THE DEVICE INDEPENDENT COMPLETION PROCESSING FOR
                        0000     31 ;       I/O PACKETS.  IT IS INVOKED BY QUEUEING THE PACKET ON THE I/O POST QUEUE
                        0000     32 ;       AND TRIGGERING THE IPL$_IOPOST SOFTWARE INTERRUPT.  SOME OF THE IOPOST
                        0000     33 ;       OPERATIONS SUCH AS SETTING EVENT FLAGS, UNLOCKING BUFFER PAGES,
                        0000     34 ;       RELEASING BUFFERS AND PAGING I/O COMPLETION ARE PERFORMED IN THE IOPOST
                        0000     35 ;       INTERRUPT SERVICE ROUTINE, WHILE OTHER OPERATIONS THAT REQUIRE ACCESS
                        0000     36 ;       TO PROCESS ADDRESS SPACE ARE PERFORMED BY SENDING A SPECIAL KERNEL AST.
                        0000     37 ;
                        0000     38 ; ENVIRONMENT:  MODE = KERNEL, RESIDENT
                        0000     39 ;
                        0000     40 ;--
                        0000     41          .SBTTL  HISTORY                 ; DETAILED
                        0000     42 ;
                        0000     43 ; AUTHOR: R. HUSTVEDT, CREATION DATE: 26-AUG-76
                        0000     44 ;
                        0000     45 ; MODIFIED BY:
                        0000      .1 ;                  Dave Butenhof           12-may-1980, Version 5.4
                        0000      .2 ;       01         Modify VMS source for supervisor
                        0000      .3 ;                  Dave Butenhof           18-jun-1980, version 5.5
                        0000      .4 ;       02         Restore buffered I/O support.
                        0000      .5 ;
                        0000      .6 ;                  Dave Butenhof   17-feb-1981, version 6.3
                        0000      .7 ;       03         Change SEP psect to CODE, correct some reference truncation
                        0000      .8 ;                  errors.
                        0000      .9 ;       04         Bob Bergazzi    6-May,1983      Version 6.11
                        0000     .10 ;                  Corrected truncation error.
                        0000     46 ;
                        0000     47 ;       V0211   KDM0103         KATHLEEN D. MORSE       13-MAR-1980
```

```
0000     48 ;                  REMOVE DUPLICATE MODIFICATION HISTORY.
0000     49 ;
0000     50 ;        V0210   RIH0059         RICHARD I. HUSTVEDT     25-FEB-1980
0000     51 ;                CHANGE PRIORITY INCREMENT CLASS FOR PAGEFAULT TO GIVE
0000     52 ;                SEPARATE AND LOWER BOOST.
0000     53 ;
0000     54 ;        V0209   KDM0094         KATHLEEN D. MORSE       22-JAN-1980     13:30
0000     55 ;                ADD CHECK FOR DRIVER RETURNING TOTAL BYTE COUNT TRANSFERRED
0000     56 ;                ON PAGE READ ERROR.  ASSUME THIS COUNT IS WRONG AND RESET
0000     57 ;                IT TO ZERO BYTES TRANSFERRED.  THIS CAUSES THE FIRST PAGE OF
0000     58 ;                THE TRANSFER TO BE THE PAGE WITH THE ERROR.
0000     59 ;
0000     60 ;        V0208   RIH0036         RICHARD I. HUSTVEDT     02-NOV-1979     09:36
0000     61 ;                HONOR IRP$V_TERMIO AND IRP$V_FILACP FLAGS FOR PERFORMANCE
0000     62 ;                IMPROVEMENT.
0000     63 ;
0000     64 ;        V0207   RIH0033         RICHARD I. HUSTVEDT     19-OCT-1979     15:32
0000     65 ;                MOVE BYTCNT FROM PCB TO JIB.
0000     66 ;
0000     67 ;        V0206   RIH0032         RICHARD I. HUSTVEDT     10-SEP-1979     11:15
0000     68 ;                ADD END ACTION DISPATCHING FOR INTERNAL PACKET COMPLETION.
0000     69 ;
0000     70 ;        V0205   KDM0054         KATHLEEN D. MORSE       31-AUG-1979     15:35
0000     71 ;                FIX INTERFACE BUG IN CALLING PMS$START_RQ.
0000     72 ;
0000     73 ;        V0204   ACG23542        ANDREW C. GOLDSTEIN     04-MAY-1979     16:41
0000     74 ;                CHECK LBN OF MAPPED VIRTUAL I/O AGAINST UCB$L_MAXBLOCK.
0000     75 ;
0000     76 ;        V0203   KDM0029         KATHLEEN D. MORSE       30-APR-1979     10:00
0000     77 ;                ADD CODE TO MAKE $UPDSEC WORK FOR SHARED MEMORY.
0000     78 ;
0000     79 ;        V0202   SRB0001         STEVE BECKHARDT         29-MAR-1979     10:00
0000     80 ;                ADDED CODE TO UNLOCK REGIONS SPECIFIED IN IRPE'S AND
0000     81 ;                TO DEALLOCATE IRPE'S.
0000     82 ;
0000     83 ;        V0201   RIH21586        R. I. HUSTVEDT          12-JAN-1979     10:00
0000     84 ;                CORRECTED BUFFER PROBING FOR BUFFERED I/O COMPLETION.
0000     85 ;
```

```
                                        0000     87          .SBTTL  DECLARATIONS
                                        0000     88 ;
                                        0000     89 ; INCLUDE FILES:
                                        0000     90 ;
                                        0000     91          $ACBDEF                         ; AST CONTROL BLOCK DEFINITIONS
                                        0000     92          $AQBDEF                         ; DEFINE AQB OFFSETS
                                        0000     93          $CADEF                          ; CONDITIONAL ASSEMBLY PARAMETERS
                                        0000     94          $CCBDEF                         ; CCB DEFINITIONS
                                        0000     95          $CXBDEF                         ; DEFINE CXB OFFSETS
                                        0000     96          $IPLDEF                         ; IPL DEFINITIONS
                                        0000     97          $IRPDEF                         ; IRP DEFINITIONS
                                        0000     98          $IRPEDEF                        ; IRPE DEFINITIONS
                                        0000     99          $JIBDEF                         ; JIB DEFINITIONS
                                        0000    100          $PCBDEF                         ; PCB DEFINITIONS
                                        0000    101          $PFNDEF                         ; PFN DATA BASE DEFINITIONS
                                        0000    102          $PHDDEF                         ; PROCESS HEADER DEFINITIONS
                                        0000    103          $PRDEF                          ; PROCESSOR REGISTER DEFINITIONS
                                        0000    104          $PRIDEF                         ; PRIORITY INCREMENT DEFS
                                        0000    105          $PTEDEF                         ; PAGE TABLE ENTRY DEFINITIONS
                                        0000    106          $RSNDEF                         ; DEFINE RESOURCE WAIT NUMBERS
                                        0000    107          $UCBDEF                         ; DEFINE UCB OFFSETS
                                        0000    108          $VADEF                          ; DEFINE VIRTUAL ADDRESS FIELDS
                                        0000    109          $VCBDEF                         ; DEFINE VCB OFFSETS
                                        0000    110          $WCBDEF                         ; DEFINE WCB OFFSETS
                                        0000    111          $WQHDEF                         ; WAIT QUEUE HEADER DEFINITIONS
                                        0000    112
                                        0000    113 ;
                                        0000    114 ; OWN STORAGE:
                                        0000    115 ;
                                    00000000     .1          .PSECT  DATA, SHR, NOEXE, NOWRT, LONG
                                        0000     .2          MODNAM  IOPOST
                    45 20 50 43 41 00' 0007     .3 AACPERR: .ASCIC  .ACP ERROR.
                          52 4F 52 52  000D
                                09     0007
                                    00000000     .4          .PSECT  CODE, SHR, EXE, LONG, NOWRT
-1                                      0000    117 PRITBL:                                 ; TABLE OF PRIORITY INCR CLASSES
                                  01    0000    118          .BYTE   PRI$_IOCOM             ; 0 => DIRECT WRITE
                                  03    0001    119          .BYTE   PRI$_TOCOM             ; 1 => BUFFERED WRITE
                                  01    0002    120          .BYTE   PRI$_IOCOM             ; 2 => DIRECT READ
                                  04    0003    121          .BYTE   PRI$_TICOM             ; 3 => BUFFERED READ
```

```
                                0004   123          .SBTTL  I/O COMPLETION POSTING
                                0004   124 ;++
                                0004   125 ; FUNCTIONAL DESCRIPTION:
                                0004   126 ;
                                0004   127 ;       IOC$IOPOST IS INITIATED BY TRIGGERING AN IPL$_IOPOST SOFTWARE
                                0004   128 ;       INTERRUPT AFTER PLACING A COMPLETED I/O PACKET IN THE IOPOST
                                0004   129 ;       QUEUE.  IOC$IOPOST PERFORMS ALL APPROPRIATE COMPLETION ACTIVITY
                                0004   130 ;       REQUIRED FOR THE PACKET EITHER DIRECTLY OR BY QUEUEING KERNEL
                                0004   131 ;       ASTS TO CONCLUDE PROCESSING IN THE CONTEXT OF THE PROCESS
                                0004   132 ;       WHEN REQUIRED.
                                0004   133 ;
                                0004   134 ; CALLING SEQUENCE:
                                0004   135 ;
                                0004   136 ;       SOFTINT #IPL$_IOPOST
                                C004   137 ;
                                0004   138 ; INPUT PARAMETERS:
                                0004   139 ;
                                0004   140 ;       NONE
                                0004   141 ;
                                0004   142 ; IMPLICIT INPUTS:
                                0004   143 ;
                                0004   144 ;       IOC$GL_PSFL - IOPOSTING QUEUE
                                0004   145 ;
                                0004   146 ; OUTPUT PARAMETERS:
                                0004   147 ;
                                0004   148 ;       NONE
                                0004   149 ;
                                0004   150 ;--
                                0004   151
                                0004   152          .ENABL  LSB
                                0004   153 IOC$IOPOST::                                ; I/O POSTING INTERRUPT
                     7E   54    7D  0004   154          MOVQ    R4,-(SP)                ; SAVE
                     7E   52    7D  0007   155          MOVQ    R2,-(SP)                ; NORMAL
                     7E   50    7D  000A   156          MOVQ    R0,-(SP)                ; REGISTERS
          00000000'FF   0F    0D  000D    .1 IOPOST:  REMQUE  @IOC$GL_PSFL,R5         ; GET HEAD OF POST QUEUE
                     55             0013
-1                   11    1C  0014   158          BVC     10$                     ; QUEUE NOT YET EMPTY
                50   8E    7D  0016   159          MOVQ    (SP)+,R0                ; RESTORE
                52   8E    7D  0019   160          MOVQ    (SP)+,R2                ; REGISTERS
                54   8E    7D  001C   161          MOVQ    (SP)+,R4                ; AND EXIT
                           02  001F   162          REI                             ; IF QUEUE EMPTY
                               0020   163
                     00C5  31  0020   164 5$:     BRW     VIRTUAL                 ; PROCESS VIRTUAL I/O COMPLETION
                               0023   165
                           61  16  0023   166 7$:     JSB     (R1)                    ; CALL END ACTION ROUTINE
                           E6  11  0025   167          BRB     IOPOST                  ;
                               0027   168
                51   0C A5    D0  0027   169 10$:     MOVL    IRP$L_PID(R5),R1        ; GET PID/END ACTION ADDRESS
                           F6  19  0028   170          BLSS    7$                      ; BR IF END ACTION ADDRESS
                               002D   171                                          ; (SYSTEM SPACE ADDRESSES ARE NEGATIVE)
                51   51    3C  002D   172          MOVZWL  R1,R1                   ; GET PROCESS INDEX
          00000000'FF41    D0  0030    .1         MOVL    @L^SCH$GL_PCBVEC[R1],R4 ; And translate to PCB address
                     54        0037
-1          03 2A A5   00    E1  0038   174          BBC     #IRP$V_BUFIO,IRP$W_STS(R5),12$  ; IF CLEAR, DIRECT I/O
                     005C  31  003D   175          BRW     BUFIO                   ; BUFFERED I/O
                  3E A4    B6  0040   176 12$:     INCW    PCB$W_DIOCNT(R4)        ; UPDATE DIRECT I/O COUNT
                53   2C A5    D0  0043   177          MOVL    IRP$L_SVAPTE(R5),R3     ; GET ADDRESS OF FIRST PTE
```

```
                                0047   178
                                0047   179          ASSUME  IRP$V_PAGIO LE 7
                                0047   180          ASSUME  IRP$V_SWAPIO LE 7
                 2A A5   44 8F   03    0047   181          BITB    #<IRP$M_PAGIO ! IRP$M_SWAPIO>,IRP$W_STS(R5) ; PAGIO OR SWAPIO?
                            37    2    004C   182          BNEQ    PAGIO_OR_SWAPIO
                                0004E  183
                                0004E  184 ;
                                0004E  185 ; DIRECT I/O COMPLETION
                                0004E  186 ;
                                0004E  187
                            53   D5   004E   188 DIRIO:  TSTL    R3                          ; PTE ADDRESS VALID?
                            29   13   0050   189          BEQL    14$                         ; IF EQL NO PAGES TO UNLOCK
                       51   32 A5   3C   0052   190          MOVZWL  IRP$W_BCNT(R5),R1           ; GET REQUESTED TRANSFER BYTE COUNT
                       52   30 A5   3C   0056   191          MOVZWL  IRP$W_BOFF(R5),R2           ; GET BYTE OFFSET IN PAGE
                    0B 2A A5    04   E1   005A   192          BBC     #IRP$V_VIRTUAL,IRP$W_STS(R5),UNLOCK ; BRANCH IF NOT VIRTUAL I/O
                         BD 34 A5   E9   005F   193          BLBC    IRP$L_IOST1(R5),5$          ; BRANCH IF ERROR IN VIRTUAL REQUEST
                    3E A5   36 A5   B1   0063   194          CMPW    IRP$L_IOST1+2(R5),IRP$W_OBCNT(R5) ; IF COMPLETED ORIGINAL BYTE COUNT
                                0068   195                                              ; THEN NO SPECIAL VIRTUAL PROCESSING
                            B6   12   0068   196          BNEQ    5$                          ; OTHERWISE DO THE SEGMENTED COMPLETION
                   51   01FF C142   9E   006A   197 UNLOCK: MOVAB   511(R1)[R2],R1           ; COMBINE OFFSET AND COUNT AND ROUND
                       51   F7 8F   78   0070   198          ASHL    #-VA$S_BYTE,R1,R1           ; CONVERT TO NUMBER OF PAGES
                            51        0074
                     00000000'EF   16   0075     .1         JSB     L^MMG$UNLOCK                ; Unlock pages
-1                  03 2A A5    0B   E1   007B   200 14$:    BBC     #IRP$V_EXTEND,IRP$W_STS(R5),15$ ; BRANCH IF NO IRPE'S ATTACHED
                          02FE   30   0080   201          BSBW    UNLOCK_MORE                 ; UNLOCK AREAS DESCRIBED IN IRPE'S
                                0083   202 15$:                                            ; REFERENCE LABEL
-7                          30   11   0083   210          BRB     30$                         ;
                                0085   211
                                0085   212 ;
                                0085   213 ;       PAGE I/O OR SWAP I/O COMPLETION
                                0085   214 ;
                                0085   215
                                0085   216 PAGIO_OR_SWAPIO:
                                0085     .1         ERRSUP_S
-39                             0096   256 ;
                                0096   257 ; BUFFERED I/O COMPLETION
                                0096   258 ;
                                0096   259
                         3A A4   B6   0096   260 BUFIO:  INCW    PCB$W_BIOCNT(R4)            ; UPDATE BUFFERED I/O COUNT
                  03 2A A5    0C   E1   0099   261          BBC     #IRP$V_FILACP,IRP$W_STS(R5),NOTACP ; BR IF NOT ACP I/O
                         3E A4   B6   009E   262          INCW    PCB$W_DIOCNT(R4)            ; RESTORE DIRECT I/O COUNT
                                00A1   263 NOTACP:                                          ;
-9                     50   2C A5   D0   00A1   273          MOVL    IRP$L_SVAPTE(R5),R0         ; ANY BUFFER SPECIFIED?
                            0E   13   00A5   274          BEQL    30$                         ; IF EQL NO
                       01E8'CF   9E   00A7   275          MOVAB   W^BUFPOST,ACB$L_KAST(R5) ; ASSUME READ FUNCTION
                         18 A5        00AB
                  09 2A A5    01   E0   00AD   276          BBS     #IRP$V_FUNC,IRP$W_STS(R5),40$  ; IF SET, READ FUNCTION
                          FF48'   30   00B2   277          BSBW    EXE$DEANONPAGED             ; DEALLOCATE WRITE BUFFER
                       02BE'CF   9E   00B5   278 30$:    MOVAB   W^DIRPOST,ACB$L_KAST(R5) ; SET SPECIAL KERNEL AST ADDRESS
                         18 A5        00B9
                         02   00   EF   00B8   279 40$:    EXTZV   #IRP$V_BUFIO,#2,IRP$W_STS(R5),R0  ; GET PACKET TYPE
                    50   2A A5        00BE
                  03 2A A5    09   E0   00C1   280          BBS     #IRP$V_TERMIO,IRP$W_STS(R5),50$ ; BR IF TERMINAL I/O
                         50   01   AA   00C6   281          BICW    #1,R0                       ; ELSE TREAT AS NORMAL I/O COMPLETION
                                00C9   282 50$:                                             ; FOR PRIORITY INCREMENT SELECTION
                    51   0C A5   D0   00C9   283          MOVL    IRP$L_PID(R5),R1            ; PROCESS IDENTIFICATION
                52   FF2E CF40   9A   00CD   284          MOVZBL  PRITB[R0],R2                ; SET PRIORITY INCREMENT CLASS
```

```
                 53   22 A5    9A  00D3   285          MOVZBL  IRP$B_EFN(R5),R3        ; GET EVENT FLAG NUMBER
                       FF26'  30  00D7   286          BSBW    SCH$POSTEF              ; AND POST IT
                              00DA   287 IOPOST_KAST:
              0B A5    80 8F   88  00DA   288          BISB    #^X80,ACB$B_RMOD(R5)    ; SET INTERNAL AST FLAG
                00000000'EF   16  00DF     1          JSB     SCH$QAST                ; NOW QUEUE THE KERNEL AST              [04]
-1                     FF25   31  00E5   290          BRW     IOPOST                  ; GET NEXT PACKET TO POST
                              00E8   291          .DSABL  LSB
                              00E8   292
```

```
                                00E8     .2          .SBTTL  VIRTUAL I/O COMPLETION
                                00E8     .3 ;
                                00E8     .4 ; VIRTUAL I/O COMPLETION
                                00E8     .5 ;
                                00E8     .6 ; CALLING SEQUENCE:
                                00E8     .7 ;
                                00E8     .8 ;       BRW     VIRTUAL
-6                              00E8    299 ;
                                00E8    300 ; INPUTS:
                                00E8    301 ;
                                00E8     .1 ;       R1 = REQUESTED BYTE COUNT, POSSIBLY DIFFERENT FROM TRANSFERRED
                                00E8     .2 ;            BYTE COUNT FOR MAGTAPE
                                00E8     .3 ;       R2 = IRP$W_BOFF CONTENTS
                                00E8     .4 ;       R3 = SVAPTE OF START OF TRANSFER
                                00E8     .5 ;
                                00E8     .6 ; OUTPUTS:
                                00E8     .7 ;
                                00E8     .8 ;       BRANCHES TO UNLOCK, PRESERVING R1,R2,R3
                                00E8     .9 ;       OR BRANCHES TO IOPOST
                                00E8    .10 ;
                                00E8    .11
                                00E8    .12          .ENABL  LSB
                                00E8    .13
                                00E8    .14 VIRTUAL:                                    ; VIRTUAL I/O FUNCTION
          50    36 A5    3C     00E8    .15          MOVZWL  IRP$L_IOST1+2(R5),R0       ; GET ACTUAL NUMBER OF BYTES TRANSFERED
          3C A5    50    A0     00EC    .16          ADDW    R0,IRP$W_ABCNT(R5)         ; ACCUMULATE TOTAL BYTES TRANSFERED
          50    F7 8F    78     00F0    .17          ASHL    #-VA$S_BYTE,R0,R0          ; CALCULATE NUMBER OF BLOCKS TRANSFERED
                   50           00F4
          40 A5    50    C0     00F5    .18          ADDL    R0,IRP$L_SEGVBN(R5)        ; CALCULATE DISK ADDRESS OF NEXT SEGMENT
       36 A5    3C A5    B0     00F9    .19          MOVW    IRP$W_ABCNT(R5),IRP$L_IOST1+2(R5) ; SET ACCUMULATED BYTES TRANSFERED
             33 34 A5    E9     00FE    .20          BLBC    IRP$L_IOST1(R5),20$        ; IF LBC I/O ERROR
          50    1C A5    D0     0102    .21          MOVL    IRP$L_UCB(R5),R0           ; GET ADDRESS OF DEVICE UCB
       1F 34 A0    00'  E0     0106    .22          BBS     S^#DEV$V_SQD,UCB$L_DEVCHAR(R0),10$ ; IF SET, SEQUENTIAL DEVICE
       3E A5    3C A5    A3     0108    .23          SUBW3   IRP$W_ABCNT(R5),IRP$W_OBCNT(R5),- ; CALCULATE BYTES REMAINING
                32 A5           0110    .24                  IRP$W_BCNT(R5)             ;
                   16    13     0112    .25          BEQL    10$                        ; IF EQL NONE
          51    F7 8F    78     0114    .26          ASHL    #-VA$S_BYTE,R1,R1          ; CALCULATE NUMBER OF PAGES REQUESTED
                   51           0118
                                0119    .27 QNXTSEG:
          2C A5    6341 DE     0119    .28          MOVAL   (R3)[R1],IRP$L_SVAPTE(R5)  ; SET ADDRESS OF NEXT PTE ENTRY
             53    55    D0     011E    .29          MOVL    R5,R3                      ; COPY I/O REQUEST PACKET ADDRESS
          55    1C A3    D0     0121    .30          MOVL    IRP$L_UCB(R3),R5           ; GET ADDRESS OF DEVICE UCB
                25    10     0125    .31          BSBB    IOC$QNXTSEG                ; QUEUE THE NEXT VIRTUAL SEGMNET
                FEE3    31     0127    .32 5$:      BRW     IOPOST
                               012A    .33 ;
                               012A    .34 ; ALL SEGMENTS OF THIS TRANSFER ARE COMPLETE
                               012A    .35 ;
          51    3E A5    3C     012A    .36 10$:     MOVZWL  IRP$W_OBCNT(R5),R1         ; GET ORIGINAL BYTE COUNT
          53    44 A5    D0     012E    .37          MOVL    IRP$L_DIAGBUF(R5),R3       ; GET ORIGINAL PAGE TABLE ADDRESS
                FF35    31     0132    .38          BRW     UNLOCK                     ;
                               0135    .39
                               0135    .40 ;
                               0135    .41 ; I/O OPERATION ENDED WITH AN UNSUCCESSFUL STATUS
                               0135    .42 ;
                               0135    .43 ;            IF THE DEVICE IS A SEQUENTIAL DEVICE, THEN THE I/O PACKET IS
                               0135    .44 ;            MERELY SENT TO THE ACP FOR NOTIFICATION OF THE ERROR.
                               0135    .45 ;
```

```
                                0135   .46 ;                  IF THE DEVICE IS A RANDOM DEVICE, THEN THE VIRTUAL BLOCK NUMBER
                                0135   .47 ;                  STORED IN IRP$L_SEQVBN IS THE BLOCK THAT HAS AN ERROR.
                                0135   .48 ;
                                0135   .49
          53    55    D0      0135   .50 20$:    MOVL    R5,R3                          ; COPY IRP ADDRESS
            3E  A4    B7      0138   .51        DECW    PCB$W_DIOCNT(R4)               ; ADJUST DIRECT I/O COUNT
       2A A3      10    AA    013B   .52        BICW    #IRP$M_VIRTUAL,IRP$W_STS(R3) ; CLEAR VIRTUAL I/O FLAG
    20 A3    44 A3    D0      013F   .53        MOVL    IRP$L_DIAGBUF(R3),IRP$L_SVAPTE(R3) ; RESET PAGE TABLE ADDRESS
       52    3E A3    3C      0144   .54        MOVZWL  IRP$W_OBCNT(R3),R2             ; GET ORIGINAL BYTE COUNT
             5B    10        0148   .55        BSBB    IOC$QTOACP                     ; QUEUE PACKET TO ACP
             DB    11        014A   .56        BRB     5$
                             014C   .57
                             014C   .58        .DSABL  LSB
```

```
                                014C    .60          .SBTTL  QUEUE NEXT SEGMENT
                                014C    .61 ;
                                014C    .62 ; FUNCTIONAL DESCRIPTION:
                                014C    .63 ;
                                014C    .64 ;       IOC$QNXTSEG PERFORMS THE FUNCTION OF QUEUEING THE NEXT
                                014C    .65 ;       SEGMENT OF A VIRTUAL I/O REQUEST THAT DID NOT MAP TO A
                                014C    .66 ;       SINGLE CONTIGUOUS I/O REQUEST.
                                014C    .67 ;
                                014C    .68 ; CALLING SEQUENCE:
                                014C    .69 ;
                                014C    .70 ;       BSBW    IOC$QNXTSEG
-289                            014C    591 ;
                                014C    592 ; INPUTS:
                                014C    593 ;
                                C14C     .1 ;       R3 = I/O REQUEST PACKET ADDRESS
                                014C     .2 ;       R4 = PCB ADDRESS ASSOCIATED WITH THE PID IN THE PACKET
                                014C     .3 ;       R5 = UCB ADDRESS OF THE ASSOCIATED DEVICE
-4                              014C    598 ;
                                014C    599 ; OUTPUTS:
                                014C    600 ;
                                014C     .1 ;       R4 NOT PRESERVED
                                014C     .2 ;
                                014C     .3 IOC$QNXTSEG::
            52    18 A3    D0   014C     .4          MOVL    IRP$L_WIND(R3),R2       ; GET ADDRESS OF MAPPING WINDOW
            51    32 A3    3C   0150     .5          MOVZWL  IRP$W_BCNT(R3),R1       ; GET SIZE OF NEXT SEGMENT
            50    40 A3    D0   0154     .6          MOVL    IRP$L_SEGVBN(R3),R0     ; GET STARTING VIRTUAL BLOCK NUMBER
                                0158     .7 ;
                                0158     .8 ; ALTERNATE ENTRY TO IOC$QNXTSEG:
                                0158     .9 ;
                                0158    .10 ;       BSBW    IOC$QNXTSEG1
                                0158    .11 ;
                                0158    .12 ; ADDITIONAL INPUTS:
                                0158    .13 ;
                                0158    .14 ;       R0 = VIRTUAL BLOCK NUMBER OF START OF NEXT SEGMENT
                                0158    .15 ;       R1 = DESIRED BYTE COUNT OF NEXT SEGMENT
                                0158    .16 ;       R2 = WINDOW ADDRESS
                                0158    .17 ;
                                0158    .18 IOC$QNXTSEG1::
          3E A4    B7   0158    .19          DECW    PCB$W_DIOCNT(R4)        ; ADJUST THE DIRECT I/O COUNT
 00000000'EF    16   015B    .20          JSB     L^IOC$MAPVBLK           ; Map virtual to logical block
  32 A3    52    A3   0161    .21          SUBW3   R2,IRP$W_BCNT(R3),IRP$W_BCNT(R3) ; CALCULATE SIZE OF NEXT SEGMENT
       32 A3          0165
          3C    13   0167    .22          BEQL    30$                     ; IF EQL TOTAL MAP FAILURE
    50    51    D0   0169    .23          MOVL    R1,R0                   ; COPY STARTING LOGICAL BLOCK NUMBER
  52    32 A3    3C   016C    .24          MOVZWL  IRP$W_BCNT(R3),R2       ; GET TRANSFER BYTE COUNT
          52    D7   0170    .25          DECL    R2                      ; ROUND DOWN AND...
  52    F7 8F    78   0172    .26          ASHL    #-VA$S_BYTE,R2,R2       ; SHIFT DOWN FOR BLOCK COUNT - 1
          52          0176
    52    51    C0   0177    .27          ADDL    R1,R2                   ; COMPUTE ENDING BLOCK NUMBER
          13    1F   017A    .28          BCS     25$                     ; BRANCH ON OVERFLOW
0084 C5   52    D1   017C    .29          CMPL    R2,UCB$L_MAXBLOCK(R5)   ; AND CHECK AGAINST DEVICE SIZE
          0C    1E   0181    .30          BGEQU   25$                     ; BRANCH IF NOT LEGAL
 00000000'EF    16   0183    .31          JSB     L^IOC$CVTLOGPHY         ; Convert logical to physical block
 00000000'EF    17   0189    .32          JMP     L^EXE$INSIOQ            ; Insert I/O packet in device queue
                             018F    .33                                  ; AND RETURN
                             018F    .34 ;
                             018F    .35 ; TO HERE IF THE VIRTUAL BLOCKS MAP OFF THE END OF THE VOLUME. COMPLETE THE
```

```
                                018F    .36 ; I/O WITH AN ERROR. WE QUEUE THE PACKET FOR PROCESSING, RATHER THAN WANDERING
                                018F    .37 ; OFF INTO THE COMPLETION CODE BECAUSE THIS IS A GENERALLY CALLABLE ROUTINE.
                                018F    .38 ;
                     0000'8F    3C   018F    .39 25$:    MOVZWL  #SS$_ILLBLKNUM,IRP$L_IOST1(R3) ; SET ILLEGAL BLOCK NUMBER STATUS
                       34 A3         0193
                       38 A3    D4   0195    .40         CLRL    IRP$L_IOST2(R3)         ; ZERO 2ND I/O STATUS LONGWORD
                          63    0E   0198    .41         INSQUE  (R3),@IOC$GL_PSBL       ; INSERT AT TAIL OF I/O POST QUEUE
                00000000'FF         019A
                          03    12   019F    .42         BNEQ    26$                     ; BRANCH IF NOT EMPTY
                                01A1    .43         SOFTINT #IPL$_IOPOST            ; WAKE UP I/O COMPLETION
                                05   01A4    .44 26$:    RSB
                                01A5    .45
                                01A5    .46 30$:
                                01A5    .47 ;
                                C1A5    .48 ; ALTERNATE ENTRY TO IOC$WAKACP:
                                01A5    .49 ;
                                01A5    .50 ;       BSBW    IOC$QTOACP
-29                             01A5   630 ;
                                01A5   631 ; INPUTS:
                                01A5   632 ;
                                01A5    .1 ;        R2 = DESIRED BYTE COUNT
                                01A5    .2 ;        R3 = IRP ADDRESS
                                01A5    .3 ;        PCB$W_DIOCNT(R4) ALREADY DECREMENTED
                                01A5    .4 ;
                                01A5    .5 IOC$QTOACP:
                       32 A3    52   B0   01A5    .6          MOVW    R2,IRP$W_BCNT(R3)       ; SET REMAINING BYTES TO TRANSFER
                       52    18 A3   D0   01A9    .7          MOVL    IRP$L_WIND(R3),R2       ; GET WINDOW ADDRESS
                       08 A2    02   E0   01AD    .8          BBS     #WCB$V_NOTFCP,WCB$B_ACCESS(R2),- ; IF SET THEN
                                20         01B1    .9                  NOTFCP%WCB              ; NOT FCP WINDOW
                       52    1C A3   D0   01B2    .10         MOVL    IRP$L_UCB(R3),R2        ; GET ADDRESS OF DEVICE UCB
                                01B6    .11 ;
                                01B6    .12 ; FUNCTIONAL DESCRIPTION:
                                01B6    .13 ;
                                01B6    .14 ;       SUBROUTINE TO QUEUE AN I/O PACKET FOR AN ACP PROCESS AND WAKE
                                01B6    .15 ;       THE PROCESS IF ITS QUEUE WAS PREVIOUSLY EMPTY.
                                01B6    .16 ;
                                01B6    .17 ; CALLING SEQUENCE:
                                01B6    .18 ;
                                01B6    .19 ;       BSBW    IOC$WAKACP
                                01B6    .20 ;
                                01B6    .21 ; INPUTS:
                                01B6    .22 ;
                                01B6    .23 ;       R2 = DEVICE UCB ADDRESS
                                01B6    .24 ;       R3 = I/O REQUEST PACKET ADDRESS
-4                              01B6   637 ;
                                01B6   638 ; OUTPUTS:
                                01B6   639 ;
                                01B6    .1 ;        R4 ALTERED
                                01B6    .2 ;
                                01B6    .3 IOC$WAKACP::                             ; QUEUE I/O PACKET AND WAKE ACP PROCESS
                                01B6    .4          DSBINT  #IPL$_SYNCH             ; SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
                                01BC    .5          BUG_CHECK NONEXSTACP            ; NONEXISTENT ACP PROCESS
                                01CE    .6 10$:     ENBINT                          ; RESTORE SAVED IPL
                                05   01D1    .7          RSB                             ;
                                01D2    .8 ;
                                01D2    .9 ; WINDOW IS NOT AN FCP WINDOW, ONLY USED FOR BOOT TIME INTIALIZED WINDOWS
                                01D2    .10 ; FOR CONTIGUOUS FILES.  IT IS NOT POSSIBLE TO NEED TO TURN SUCH A WINDOW.
```

```
                        01D2    .11 ;
                        01D2    .12 NOTFCPWCB:
                        01D2    .13         BUG_CHECK NOTFCPWCB,FATAL
```

```
                                01E8   .15          .SBTTL  BUFFERED READ COMPLETION AST ROUTINE
                                01E8   .16 ;++
                                01E8   .17 ; FUNCTIONAL DESCRIPTION:
                                01E8   .18 ;
                                01E8   .19 ;       BUFPOST PERFORMS ALL NECESSARY COMPLETION OPERATIONS REQUIRED
                                01E8   .20 ;       FOR A BUFFERED READ OPERATION IN THE CONTEXT OF THE PROCESS
                                01E8   .21 ;       ISSUING THE I/O REQUEST.
-59                             01E8   699 ;
                                01E8   700 ; CALLING SEQUENCE:
                                01E8   701 ;
                                01E8    .1 ;       JSB     BUFPOST
                                01E8    .2 ;
                                01E8    .3 ; INPUT PARAMETERS:
                                01E8    .4 ;
                                01E8    .5 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                01E8    .6 ;       R5 = IRP/AST CONTROL BLOCK.
                                01E8    .7 ;
                                01E8    .8 ; IMPLICIT INPUTS:
                                01E8    .9 ;
                                01E8   .10 ;       SCH$GL_CURPCB - POINTER TO PCB OF CURRENT PROCESS
                                01E8   .11 ;--
                                01E8   .12
                                01E8   .13 BUFPOST:                                     ; BUFFERED READ COMPLETION
          00E0 8F    BB         01E8   .14          PUSHR   #^M<R5,R6,R7>               ; SAVE REGISTERS
     56     2C A5    D0         01EC   .15          MOVL    IRP$L_SVAPTE(R5),R6         ; GET ADDRESS OF I/O BUFFER
     57     32 A5    3C         01F0   .16          MOVZWL  IRP$W_BCNT(R5),R7           ; GET COUNT OF BYTES OR DESCRIPTORS
   4A 2A A5    03    E1         01F4   .17          BBC     #IRP$V_COMPLX,IRP$W_STS(R5),40$ ;IF CLR, NOT COMPLEX BUFFER FORMAT
   4D 2A A5    05    E0         01F9   .18          BBS     #IRP$V_CHAINED,IRP$W_STS(R5),50$ ;IF SET ,CHAINED BUFFERS
        56     66    D0         01FE   .19          MOVL    (R6),R6                     ; GET ADDRESS OF FIRST BUFFER DESCRIPTOR
     50     02 A6    3C         0201   .20 10$:     MOVZWL  2(R6),R0                    ; GET COUNT OF BYTES TO TRANSFER
                32    13         0205   .21          BEQL    30$                         ; IF EQL NONE THIS DESCRIPTOR
     51     04 A6    D0         0207   .22          MOVL    4(R6),R1                    ; GET ADDRESS OF USER BUFFER
        50     51    C0         020B   .23          ADDL    R1,R0                       ; CALCULATE ENDING ADDRESS OF BUFFER
   51   01FF 8F    AA         020E   .24          BICW    #VA$M_BYTE,R1               ; TRUNCATE ADDRESS TO PAGE BOUNDARY
        50     51    C2         0213   .25          SUBL    R1,R0                       ; COMPUTE NUMBER OF BYTES TO PROBE
        54     66    3C         0216   .26          MOVZWL  (R6),R4                     ; GET OFFSET TO DATA AREA
   53   FE00 8F    32         0219   .27          CVTWL   #-^X200,R3                  ; SET ADDITION CONSTANT
                             021E   .28 20$:     IFNOWRT R0,(R1),35$,(R6)[R4]        ; CAN BUFFER BE WRITTEN?
        51     53    C2         0225   .29          SUBL    R3,R1                       ; UPDATE ADDRESS OF BUFFER
     50   6043    3E         0228   .30          MOVAW   (R0)[R3],R0                 ; UPDATE REMAINING LENGTH
             F0    14         022C   .31          BGTR    20$                         ; IF GEQ MORE TO CHECK
          02 A6    28         022E   .32          MOVC    2(R6),1(R6)[R4],@4(R6)      ; MOVE DATA TO USER BUFFER
        01 A644             0231
          04 B6             0234
        55     6E    D0         0236   .33          MOVL    (SP),R5                     ; RESTORE ADDRESS OF I/O PACKET
        56     08    C0         0239   .34 30$:     ADDL    #8,R6                       ; ADVANCE TO NEXT BUFFER DESCRIPTOR
          C2 57    F5         023C   .35          SOBGTR  R7,10$                      ; ANY MORE DESCRIPTORS TO PROCESS?
             72    11         023F   .36          BRB     130$                        ;
             6D    11         0241   .37 35$:     BRB     120$                        ; CONTINUE
        0103    30         0243   .38 40$:     BSBW    MOVBUF                      ; MOVE BUFFER TO USER
        55     6E    D0         0246   .39          MOVL    (SP),R5                     ; RETRIEVE ADDRESS OF I/O PACKET
             68    11         0249   .40          BRB     130$                        ;
        50     57    D0         024B   .41 50$:     MOVL    R7,R0                       ; SET LENGTH OF USER BUFFER
     51     04 A6    D0         024E   .42          MOVL    4(R6),R1                    ; SET ADDRESS OF USER BUFFER
        50     51    C0         0252   .43          ADDL    R1,R0                       ; CALCULATE END OF USER BUFFER
   51   01FF 8F    AA         0255   .44          BICW    #VA$M_BYTE,R1               ; TRUNCATE TO PAGE BOUNDARY
        50     51    C2         025A   .45          SUBL    R1,R0                       ; COMPUTE NUMBER OF BYTES TO PROBE
```

```
                02   00    EF   025D   .46          EXTZV   #0,#2,IRP$B_RMOD(R5),R2 ; GET REQUEST ACCESS MODE
             52   0B A5          0260
          53   FF00 8F    32   0263   .47          CVTWL   #-^X200,R3              ; SET ADDITION CONSTANT
                                0268   .48 60$:     IFNOWRT R0,(R1),90$,R2          ; CAN BUFFER BE WRITTEN?
                51   53    C2   026E   .49          SUBL    R3,R1                   ; UPDATE ADDRESS OF BUFFER
              50   6043    3E   0271   .50          MOVAW   (R0)[R3],R0             ; CALCULATE NEW LENGTH
                     F1    14   0275   .51          BGTR    60$                     ; IF GEQ MORE TO PROBE
             53   04 A6    D0   0277   .52          MOVL    4(R6),R3                ; GET STARTING ADDRESS OF USER BUFFER
          0C A6   57    B1   027B   .53 70$:     CMPW    R7,CXB$W_LENGTH(R6)     ; REMAINING LENGTH LARGER THAN DATA AREA?
                     04    1E   027F   .54          BGEQU   80$                     ; IF GEQU YES
          0C A6   57    B0   0281   .55          MOVW    R7,CXB$W_LENGTH(R6)     ; TRUNCATE LENGTH OF DATA AREA
          96   0C A6    28   0285   .56 80$:     MOVC    CXB$W_LENGTH(R6),@(R6)+,(R3) ; MOVE DATA TO USER BUFFER
                     63          0289
             55   6E    D0   028A   .57          MOVL    (SP),R5                 ; RETRIEVE ADDRESS OF I/O PACKET
          57   08 A6    A2   028D   .58          SUBW    CXB$W_LENGTH-4(R6),R7   ; REDUCE REMAINING BYTES TO TRANSFER
                     0B    13   0291   .59          BEQL    100$                    ; IF EQL DONE
          56   0C A6    D0   0293   .60          MOVL    CXB$L_LINK-4(R6),R6     ; GET ADDRESS OF NEXT BUFFER IN CHAIN
                     E2    12   0297   .61          BNEQ    70$                     ; IF NEQ MORE TO GO
                     03    11   0299   .62          BRB     100$                    ;
                   00DE    30   029B   .63 90$:     BSBW    ACCVIO                  ; SET ACCESS VIOLATION
          50   2C A5    D0   029E   .64 100$:    MOVL    IRP$L_SVAPTE(R5),R0     ; GET ADDRESS OF FIRST BUFFER
          56   10 A0    D0   02A2   .65 110$:    MOVL    CXB$L_LINK(R0),R6       ; GET ADDRESS OF NEXT BUFFER
                     0F    13   02A6   .66          BEQL    140$                    ; IF EQL NONE
                FD55'    30   02A8   .67          BSBW    EXE$DEANONPAGED         ; DEALLOCATE BUFFER
             50   56    D0   02AB   .68          MOVL    R6,R0                   ; SET ADDRESS OF NEXT BUFFER
                     F2    11   02AE   .69          BRB     110$                    ;
                   00C9    30   02B0   .70 120$:    BSBW    ACCVIO                  ; SET ACCESS VIOLATION STATUS
          50   2C A5    D0   02B3   .71 130$:    MOVL    IRP$L_SVAPTE(R5),R0     ; GET ADDRESS OF BUFFER TO RELEASE
                FD46'    30   02B7   .72 140$:    BSBW    EXE$DEANONPAGED         ; DEALLOCATE BUFFER
             00E0 8F    BA   02BA   .73          POPR    #^M<R5,R6,R7>           ; RESTORE REGISTERS
```

```
                                02BE    .75          .SBTTL  DIRECT I/O COMPLETION AST ROUTINE
                                02BE    .76 ;++
                                02BE    .77 ; FUNCTIONAL DESCRIPTION:
                                02BE    .78 ;
                                02BE    .79 ;       DIRPOST PERFORMS ALL GENERAL I/O COMPLETION ACTIVITIES WHICH
                                02BE    .80 ;       MUST BE DONE IN THE CONTEXT OF THE PROCESS. THESE INCLUDE
                                02BE    .81 ;       I/O STATUS POSTING IF AN IOSB WAS SPECIFIED, CHANNEL CONTROL
                                02BE    .82 ;       BLOCK ACTIVITY COUNT DECREMENTING, QUEUEING OF ANY REQUESTED
                                02BE    .83 ;       AST OR RELEASE OF THE I/O REQUEST PACKET.
-78                             02BE    780 ;
                                02BE    781 ; CALLING SEQUENCE:
                                02BE    782 ;
                                02BE     .1 ;       JSB     DIRPOST
-41                             02BE    824 ;
                                C2BE    825 ; INPUT PARAMETERS:
                                02BE    826 ;
                                02BE    827 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                02BE     .1 ;       R5 = IRP/AST CONTROL BLOCK ADDRESS.
-1                              02BE    829 ;
                                02BE    830 ; IMPLICIT INPUTS:
                                02BE    831 ;
                                02BE     .1 ;       SCH$GL_CURPCB - POINTER TO CURRENT PCB
                                02BE     .2 ;--
                                02BE     .3
                                02BE     .4 DIRPOST:                                     ; DIRECT I/O POSTING AST
             1C 2A A5   07   E1 02BE     .5          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R5),10$ ; IF CLR, NO DIAGNOSTIC BUFFER
                 00E0 8F   BB   02C3     .6          PUSHR   #^M<R5,R6,R7>           ; SAVE REGISTERS
               56   44 A5   D0   02C7     .7          MOVL    IRP$L_DIAGBUF(R5),R6    ; GET ADDRESS OF DIAGNOSTIC BUFFER
               57   08 A6   3C   02CB     .8          MOVZWL  IRP$W_SIZE(R6),R7       ; GET SIZE OF DIAGNOSTIC BUFFER
                  57   0C   C2   02CF     .9          SUBL    #12,R7                  ; REDUCE BY SIZE OF BUFFER HEADER
                       75   10   02D2    .10          BSBB    MOVBUF                  ; MOVE DIAGNOSTIC INFORMATION TO USER
                 00E0 8F   BA   02D4    .11          POPR    #^M<R5,R6,R7>           ; RESTORE REGISTERS
               50   44 A5   D0   02D8    .12          MOVL    IRP$L_DIAGBUF(R5),R0    ; RETRIEVE ADDRESS OF DIAGNOSTIC BUFFER
                    FD21'   30   02DC    .13          BSBW    EXE$DEANONPAGED         ; DEALLOCATE DIAGNOSTIC BUFFER
               50   28 A5   32   02DF    .14 10$:     CVTWL   IRP$W_CHAN(R5),R0       ; GET CHANNEL NUMBER (NEGATED)
         00000000'FF40   9E   02E3    .15          MOVAB   @CTL$GL_CCBBASE[R0],R1  ; SET CCB BASE ADDRESS
                       51   02EA
                 0A A1   B7   02EB    .16          DECW    CCB$W_IOC(R1)           ; DECREMENT I/O COUNT FOR CHANNEL
                       12   12   02EE    .17          BNEQ    30$                     ; NOT IDLE YET
               53   0C A1   D0   02F0    .18          MOVL    CCB$L_DIRP(R1),R3       ; GET ADDRESS OF DEACCESS PACKET
                       0C   13   02F4    .19          BEQL    30$                     ; IF EQL NONE
                 0C A1   D4   02F6    .20          CLRL    CCB$L_DIRP(R1)          ; CLEAR ADDRESS OF DEACCESS PACKET
                 0A A1   B6   02F9    .21          INCW    CCB$W_IOC(R1)           ; ACCOUNT FOR DEACCESS
               52   61   D0   02FC    .22          MOVL    CCB$L_UCB(R1),R2        ; GET ASSIGNED DEVICE UCB ADDRESS
                    FEB4   30   02FF    .23          BSBW    IOC$WAKACP              ; QUEUE I/O PACKET AND WAKE ACP
                                0302    .24                                          ; R4 ALTERERED
                                0302    .25 30$:
                                0302    .26 ;
                                0302    .27 ; R4 DOES NOT NECESSARILY HAVE CURRENT PCB ADDRESS IN IT AT THIS POINT
                                0302    .28 ;
                                0302    .29 IOC$DIRPOST1::
               50   24 A5   D0   0302    .30          MOVL    IRP$L_IOSB(R5),R0       ; GET IOSB ADDRESS
                       10   13   0306    .31          BEQL    35$                     ; IF EQL NONE SPECIFIED
                  02   00   EF   0308    .32          EXTZV   #0,#2,IRP$B_RMOD(R5),R1 ; GET REQUEST ACCESS MODE
               51   0B A5          0308
                                030E    .33          IFNOWRT #8,(R0),35$,R1          ; CAN I/O STATUS BE WRITTEN?
               60   34 A5   7D   0314    .34          MOVQ    IRP$L_IOST1(R5),(R0)    ; MOVE STATUS INTO IOSB
```

```
                13 2A A5    0B   E0  0318   .35 35$:    BBS     #IRP$V_EXTEND,IRP$W_STS(R5),50$ ; BRANCH TO DEALLOCATE IRPE'S
                08 0B A5    06   E1  031D   .36 37$:    BBC     #ACB$V_QUOTA,IRP$B_RMOD(R5),40$ ; IF CLR, NO AST SPECIFIED
                           52    D4  0322   .37          CLRL    R2                              ; SET NULL PRIORITY INCREMENT
                  00000000'EF    17  0324   .38          JMP     SCH$QAST                        ; QUEUE AST FOR REQUESTOR
                       50   55   D0  032A   .39 40$:     MOVL    R5,R0                           ; SETUP ADDRESS FOR DEALLOCATE
                          FCD0'  31  032D   .40          BRW     EXE$DEANONPAGED                 ; AND RELEASE I/O PACKET
                                 0330   .41
                                 0330   .42
                                 0330   .43          ;
                                 0330   .44          ; DEALLOCATE IRPE'S
                                 0330   .45          ;
                                 0330   .46
                   50   4C A5    D0  0330   .47 50$:     MOVL    IRP$L_EXTEND(R5),R0             ; GET ADDRESS OF FIRST IRPE
                                 0334   .48
                           54    D4  C334   .49 60$:     CLRL    R4                              ; WILL HOLD ADDRESS OF NEXT IRPE
                04 2A A0    0B   E1  0336   .50          BBC     #IRPE$V_EXTEND,IRPE$W_STS(R0),70$ ; BR. IF NO MORE IRPE'S
                   54   4C A0    D0  033B   .51          MOVL    IRPE$L_EXTEND(R0),R4            ; SAVE ADDRESS OF NEXT IRPE
                          FCBE'  30  033F   .52 70$:     BSBW    EXE$DEANONPAGED                 ; DEALLOCATE IRPE POINTED TO BY R0
                       50   54   D0  0342   .53          MOVL    R4,R0                           ; PUT ADDRESS OF NEXT IRPE IN R0
                           ED    12  0345   .54          BNEQ    60$                             ; BR. IF THERE IS ANOTHER IRPE
                           D4    11  0347   .55          BRB     37$                             ; DONE DEALLOCATING IRPE'S
```

```
                                0349   .57          .SBTTL  MOVE DATA TO USER BUFFER
                                0349   .58 ;
                                0349   .59 ; SUBROUTINE TO MOVE DATA FROM A SIMPLE BUFFERED I/O BUFFER TO A USER BUFFER
                                0349   .60 ;
                                0349   .61
                                0349   .62 MOVBUF:                                          ; MOVE BUFFER
        50    57    D0          0349   .63          MOVL    R7,R0                         ; SET LENGTH OF USER BUFFER
              2D    13          034C   .64          BEQL    15$                           ; BR IF NULL STRING
   51    04 A6    D0          034E   .65          MOVL    4(R6),R1                      ; GET ADDRESS OF USER BUFFER
        50    51    C0          0352   .66          ADDL    R1,R0                         ; CALCULATE ENDING BUFFER ADDRESS
  51    01FF 8F    AA          0355   .67          BICW    #VA$M_BYTE,R1                 ; TRUNCATE ADDRESS TO START OF PAGE
        50    51    C2          035A   .68          SUBL    R1,R0                         ; CALCULATE LENGTH OF BUFFER TO PROBE
        02    00    EF          035D   .69          EXTZV   #0,#2,IRP$B_RMOD(R5),R2       ; GET REQUEST ACCESS MODE
   52    0B A5                   0360
  53    FE00 8F    32          0363   .70          CVTWL   #-^X200,R3                    ; SET ADDITION CONSTANT
                                0368   .71 10$:     IFNOWRT R0,(R1),ACCVIO,R2             ; CAN BUFFER BE WRITTEN?
        51    53    C2          036E   .72          SUBL    R3,R1                         ; UPDATE BUFFER ADDRESS
     50    6043    3E          0371   .73          MOVAW   (R0)[R3],R0                   ; UPDATE REMAINING LENGTH OF BUFFER
              F1    14          0375   .74          BGTR    10$                           ; IF GEQ MORE TO CHECK
  96    96    57    28          0377   .75          MOVC    R7,@(R6)+,@(R6)+              ; MOVE DATA TO USER BUFFER
                    05          037B   .76 15$:     RSB                                   ;
    34 A5    00'    B0          037C   .77 ACCVIO:  MOVW    S^#SS$_ACCVIO,IRP$L_IOST1(R5) ; SET FINAL TRANSFER STATUS
                    05          0380   .78          RSB                                   ;
```

```
                                0381    .80         .SBTTL  UNLOCK AREAS IN IRPE'S
-69                             0381    901 ;++
                                0381    902 ; FUNCTIONAL DESCRIPTION:
                                0381    903 ;
                                0381     .1 ;       THIS ROUTINE UNLOCKS THE AREAS DESCRIBED BY FIELDS IN THE IRPE'S.  EACH
                                0381     .2 ;       IRPE HAS SPACE TO HOLD TWO AREA DESCRIPTIONS.
-5                              0381    909 ;
                                0381    910 ; CALLING SEQUENCE:
                                0381    911 ;
                                0381     .1 ;       BSBW    UNLOCK_MORE
-1                              0381    913 ;
                                0381    914 ; INPUT PARAMETERS:
                                0381    915 ;
-105                            0381   1021 ;       R5 = I/O REQUEST PACKET ADDRESS
                                C381   1022 ;
                                0381   1023 ; SIDE EFFECTS:
                                0381   1024 ;
                                0381   1025 ;       R0 - R3 ARE NOT PRESERVED
                                0381   1026 ;--
                                0381   1027
                                0381   1028         ASSUME  IRP$L_EXTEND EQ IRPE$L_EXTEND
                                0381   1029
                                0381   1030 UNLOCK_MORE:
                         55  DD 0381   1031         PUSHL   R5                      ; SAVE IRP ADDRESS
                                0383   1032
                                0383   1033 10$:    ; UNLOCK AREAS SPECIFIED IN NEXT IRPE
                                0383   1034
                      55  4C A5  D0 0383   1035         MOVL    IRPE$L_EXTEND(R5),R5    ; GET ADDRESS OF NEXT IRPE
                      53  2C A5  D0 0387   1036         MOVL    IRPE$L_SVAPTE1(R5),R3   ; GET SVAPTE OF FIRST AREA
                          0A  13  038B   1037         BEQL    20$                     ; BR. IF NOTHING TO UNLOCK
                      52  30 A5  3C 038D   1038         MOVZWL  IRPE$W_BOFF1(R5),R2     ; GET BYTE OFFSET IN PAGE
                      51  34 A5  D0 0391   1039         MOVL    IRPE$L_BCNT1(R5),R1     ; GET SIZE OF AREA
                          19  10  0395   1040         BSBB    UNLK                    ; UNLOCK FIRST AREA
                                0397   1041
                      53  38 A5  D0 0397   1042 20$:    MOVL    IRPE$L_SVAPTE2(R5),R3   ; GET SVAPTE OF SECOND AREA
                          0A  13  039B   1043         BEQL    30$                     ; BR. IF NOTHING TO UNLOCK
                      52  3C A5  3C 039D   1044         MOVZWL  IRPE$W_BOFF2(R5),R2     ; GET BYTE OFFSET IN PAGE
                      51  40 A5  D0 03A1   1045         MOVL    IRPE$L_BCNT2(R5),R1     ; GET SIZE OF AREA
                          09  10  03A5   1046         BSBB    UNLK                    ; UNLOCK SECOND AREA
                                03A7   1047
                   D7 2A A5   0B  E0 03A7   1048 30$:    BBS     #IRPE$V_EXTEND,IRPE$W_STS(R5),10$ ; BR. IF THERE'S ANOTHER IRPE
                         55 8ED0  03AC   1049         POPL    R5                      ; RESTORE R5
                              05  03AF   1050         RSB
                                03B0   1051
                                03B0   1052 ;
                                03B0   1053 ; LOCAL SUBROUTINE TO UNLOCK PAGES
                                03B0   1054 ;
                                03B0   1055 ;       R1 = BYTE COUNT (OR SIZE OF AREA)
                                03B0   1056 ;       R2 = BYTE OFFSET IN PAGE
                                03B0   1057 ;       R3 = SVAPTE OF START OF AREA
                                03B0   1058 ;
                                03B0   1059
                51  01FF C142   9E 03B0   1060 UNLK:    MOVAB   511(R1)[R2],R1          ; COMBINE OFFSET AND SIZE AND ROUND
                    51   F7 8F  78 03B6   1061         ASHL    #-VA$S_BYTE,R1,R1       ; CONVERT TO NUMBER OF PAGES TO UNLOCK
                             51    03BA
                00000000'EF   16  03BB      .1         JSB     L^MMG$UNLOCK            ; Unlock pages
-1                            05  03C1   1063         RSB
```

B 1 *** GTCHAN Get channel informa
C 1 GET I/O CHANNEL DEVICE INFORMA
D 1 GET I/O CHANNEL DEVICE INFORMA
E 1 Symbol table
F 1 Symbol table
G 1 Cross reference
H 1 Cross reference
I 1 *** HELP Handle HELP command
J 1 *** HELP Handle HELP command
K 1 *** HELP Handle HELP command
L 1 *** HELP Handle HELP command
M 1 *** HELP Handle HELP command
N 1 *** HELP Handle HELP command
B 2 *** HELP Handle HELP command
C 2 *** HELP Handle HELP command
D 2 Help control routine
E 2 Help control routine
F 2 Help control routine
G 2 Help control routine
H 2 Help control routine
I 2 Help control routine
J 2 Help control routine
K 2 Help control routine
L 2 do help stuff
M 2 do help stuff
N 2 do help stuff
B 3 do help stuff
C 3 do help stuff
D 3 do help stuff
E 3 do help stuff
F 3 do help stuff
G 3 do help stuff
H 3 check for key on line
I 3 check for key on line
J 3 check for key on line
K 3 check for key on line
L 3 print key list
M 3 print key list
N 3 print help text
B 4 print help text
C 4 print help text
D 4 List other keys
E 4 List other keys
F 4 PrintHeader
G 4 PrintHeader
H 4 print_otherhelp main code
I 4 print_otherhelp main code
J 4 print_otherhelp main code
K 4 print_otherhelp main code
L 4 Read random record
M 4 Read random record
N 4 Read random record
B 5 compare keys
C 5 compare keys
D 5 compare keys
E 5 compare keys
F 5 compare keys
G 5 skip_blanks
H 5 skip_blanks
I 5 scan_word
J 5 scan_word
K 5 scan_word
L 5 find character
M 5 find character
N 5 print blank line
B 6 print blank line
C 6 - Interpreted code for TSP
D 6 - Interpreted code for TSP
E 6 - Interpreted code for TSP
F 6 DECLARATIONS
G 6 DECLARATIONS
H 6 Device description database
I 6 Device description database
J 6 Device description database
K 6 Device description database
L 6 Device description database
M 6 Device description database
N 6 Device description database
B 7 Device description database
C 7 Device description database
D 7 Device description database
E 7 Device description database
F 7 Device description database
G 7 Device description database
H 7 Device description database
I 7 Device description database
J 7 Device description database
K 7 Device description database
L 7 Device description database
M 7 Device description database
N 7 Device description database
B 8 Device description database
C 8 Device description database
D 8 Device description database
E 8 Device description database
F 8 Device description database
G 8 Device description database
H 8 Device description database
I 8 Device description database
J 8 Device description database
K 8 Device description database
L 8 Device description database
M 8 Device description database
N 8 Device description database
B 9 Device description database
C 9 Device description database
D 9 Device description database
E 9 Device description database
F 9 Device description database
G 9 Device description database
H 9 Device description database
I 9 Device description database
J 9 Device description database
K 9 Device description database
L 9 Device description database
M 9 Device description database
N 9 Device description database
B 10 Device description database
C 10 Device description database
D 10 Device description database
E 10 Device description database
F 10 Device description database
G 10 Device description database
H 10 Device description database
I 10 Symbol table
J 10 Symbol table
K 10 Symbol table
L 10 Symbol table
M 10 Symbol table
N 10 Psect synopsis
B 11 Cross reference
C 11 Cross reference
D 11 Cross reference
E 11 Cross reference
F 11 Cross reference
G 11 Cross reference
H 11 Cross reference
I 11 Cross reference
J 11 Cross reference
K 11 Cross reference
L 11 Cross reference
M 11 *** IOBASE I/O data base
N 11 *** IOBASE I/O data base
B 12 *** IOBASE I/O data base
C 12 *** IOBASE I/O data base
D 12 DECLARATIONS
E 12 DECLARATIONS
F 12 DECLARATIONS
G 12 DECLARATIONS
H 12 Device description database
I 12 Device description database
J 12 Device description database
K 12 Device description database
L 12 Device description database
M 12 Device description database
N 12 Symbol table
B 13 Symbol table
C 13 Symbol table
D 13 Symbol table
E 13 Symbol table
F 13 Psect synopsis
G 13 Cross reference
H 13 Cross reference
I 13 Cross reference
J 13 Cross reference
K 13 Cross reference
L 13 Cross reference
M 13 Cross reference
N 13 Cross reference
B 14 Cross reference
C 14 Cross reference
D 14 Cross reference
E 14 Cross reference
F 14 Cross reference
G 14 Cross reference
H 14 Cross reference
I 14 Cross reference
J 14 Cross reference
K 14 Cross reference
L 14 Cross reference

```
M 14  Cross reference
N 14  Cross reference
B 15  Cross reference
C 15  Cross reference
D 15  Cross reference
E 15  Cross reference
F 15  Cross reference
G 15  Cross reference
H 15  *** IOPOST I/O completion post
I 15  *** IOPOST I/O completion post
J 15  HISTORY    ; DETAILED
K 15  DECLARATIONS
L 15  I/O COMPLETION POSTING
M 15  I/O COMPLETION POSTING
N 15  I/O COMPLETION POSTING
B 16  VIRTUAL I/O COMPLETION
C 16  VIRTUAL I/O COMPLETION
D 16  QUEUE NEXT SEGMENT
E 16  QUEUE NEXT SEGMENT
F 16  QUEUE NEXT SEGMENT
G 16  BUFFERED READ COMPLETION AST R
H 16  BUFFERED READ COMPLETION AST R
I 16  DIRECT I/O COMPLETION AST ROUT
J 16  DIRECT I/O COMPLETION AST ROUT
K 16  MOVE DATA TO USER BUFFER
L 16  UNLOCK AREAS IN IRPE'S
```

```
                03C2  1064
                03C2  1065
                03C2  1066          .END
```

```
$ER                             =  00000002     D
$MODULE                            00000000 R   D   02
AACPERR                            00000007 R   D   02
ACB$B_RMOD                         0000000B     D
ACB$B_TYPE                         0000000A     D
ACB$C_LENGTH                       00000018     D
ACB$K_LENGTH                       00000018     D
ACB$L_AST                          00000010     D
ACB$L_ASTPRM                       00000014     D
ACB$L_ASTQBL                       00000004     D
ACB$L_ASTQFL                       00000000     D
ACB$L_KAST                         00000018     D
ACB$L_PID                          0000000C     D
ACB$V_QUOTA                     =  00000006     D
ACB$W_SIZE                         0000C008     D
ACCVIO                             0000037C R   D   03
BUFIO                              00000096 R   D   03
BUFPOST                            000001E8 R   D   03
BUG$CHECK                          ********       X   03
CCB$B_AMOD                         00000009     D
CCB$B_STS                          00000008     D
CCB$C_LENGTH                       00000010     D
CCB$K_LENGTH                       00000010     D
CCB$L_DIRP                         0000000C     D
CCB$L_UCB                          00000000     D
CCB$L_WIND                         00000004     D
CCB$W_IOC                          0000000A     D
CTL$GL_CCBBASE                     ********       X   03
CXB$L_LINK                      =  00000010     D
CXB$W_LENGTH                    =  0000000C     D
DEV$V_SQD                          ********       X   03
DIRIO                              0000004E R   D   03
DIRPOST                            000002BE R   D   03
DS_ERRSUP                          ********       X   03
EXE$DEANONPAGED                    ********       X   03
EXE$INSIOQ                         ********       X   03
IOC$CVTLOGPHY                      ********       X   03
IOC$DIRPOST1                       00000302 RG  D   03
IOC$GL_PSBL                        ********       X   03
IOC$GL_PSFL                        ********       X   03
IOC$IOPOST                         00000004 RG  D   03
IOC$MAPVBLK                        ********       X   03
IOC$QNXTSEG                        0000014C RG  D   03
IOC$QNXTSEG1                       00000158 RG  D   03
IOC$QTOACP                         000001A5 R   D   03
IOC$WAKACP                         000001B6 RG  D   03
IOPOST                             0000000D R   D   03
IOPOST_KAST                        000000DA R   D   03
IPL$_IOPOST                     =  00000004     D
IPL$_SYNCH                      =  00000007     D
IRP$B_CARCON                       00000038     D
IRP$B_EFN                          00000022     D
IRP$B_PRI                          00000023     D
IRP$B_RMOD                         0000000B     D
IRP$B_TYPE                         0000000A     D
IRP$C_LENGTH                       0000005C     D
IRP$K_LENGTH                       0000005C     D
IRP$L_ARB                          00000050     D
IRP$L_AST                          00000010     D
IRP$L_ASTPRM                       00000014     D
IRP$L_DIAGBUF                      00000044     D
IRP$L_EXTEND                       0000004C     D
IRP$L_IOQBL                        00000004     D
IRP$L_IOQFL                        00000000     D
IRP$L_IOSB                         00000024     D
IRP$L_IOST1                        00000034     D
IRP$L_IOST2                        00000038     D
IRP$L_MEDIA                        00000034     D
IRP$L_PID                          0000000C     D
IRP$L_SEGVBN                       00000040     D
IRP$L_SEQNUM                       00000048     D
IRP$L_SVAPTE                       0000002C     D
IRP$L_TT_TERM                      00000038     D
IRP$L_UCB                          0000001C     D
IRP$L_WIND                         00000018     D
IRP$M_PAGIO                     =  00000004     D
IRP$M_SWAPIO                    =  00000040     D
IRP$M_VIRTUAL                   =  00000010     D
IRP$Q_NT_PRVMSK                    0000003C     D
IRP$V_BUFIO                     =  00000000     D
IRP$V_CHAINED                   =  00000005     D
IRP$V_COMPLX                    =  00000003     D
IRP$V_DIAGBUF                   =  00000007     D
IRP$V_EXTEND                    =  0000000B     D
IRP$V_FILACP                    =  0000000C     D
IRP$V_FUNC                      =  00000001     D
IRP$V_PAGIO                     =  00000002     D
IRP$V_SWAPIO                    =  00000006     D
IRP$V_TERMIO                    =  00000009     D
IRP$V_VIRTUAL                   =  00000004     D
IRP$W_ABCNT                        0000C03C     D
IRP$W_BCNT                         00000032     D
IRP$W_BOFF                         00000030     D
IRP$W_CHAN                         00000028     D
IRP$W_FUNC                         00000020     D
IRP$W_OBCNT                        0000003E     D
IRP$W_SIZE                         00000008     D
IRP$W_STS                          0000002A     D
IRP$W_TT_PRMPT                     0000003C     D
IRPE$B_TYPE                        0000000A     D
IRPE$C_LENGTH                      00000050     D
IRPE$K_LENGTH                      00000050     D
IRPE$L_BCNT1                       00000034     D
IRPE$L_BCNT2                       00000040     D
IRPE$L_EXTEND                      0000004C     D
IRPE$L_SVAPTE1                     0000002C     D
IRPE$L_SVAPTE2                     00000038     D
IRPE$V_EXTEND                   =  0000000B     D
IRPE$W_BOFF1                       00000030     D
IRPE$W_BOFF2                       0000003C     D
IRPE$W_SIZE                        00000008     D
IRPE$W_STS                         0000002A     D
MMG$UNLOCK                         ********       X   03
MOVBUF                             00000349 R   D   03
```

```
NOTACP                          000000A1 R   D  03      SCH$POSTEF                      ********   X   03
NOTFCPWCB                       000001D2 R   D  03      SCH$QAST                        ********   X   03
PAGIO_OR_SWAPIO                 00000085 R   D  03      SIZ...                        = 00000001     D
PCB$B_ASTACT                    0000000C     D          SS$_ACCVIO                      ********   X   03
PCB$B_ASTEN                     0000000D     D          SS$_ILLBLKNUM                   ********   X   03
PCB$B_PRI                       0000000B     D          UCB$B_AMOD                      00000053     D
PCB$B_PRIB                      0000002F     D          UCB$B_CEX                       00000077     D
PCB$B_TYPE                      0000000A     D          UCB$B_CM1                       0000004A     D
PCB$B_WEFC                      0000002E     D          UCB$B_CM2                       0000004B     D
PCB$C_LENGTH                    0000008C     D          UCB$B_DEVCLASS                  00000038     D
PCB$K_LENGTH                    0000008C     D          UCB$B_DEVTYPE                   00000039     D
PCB$L_ARB                       00000084     D          UCB$B_DIPL                      00000052     D
PCB$L_ASTQBL                    00000014     D          UCB$B_DX_SCTCNT                 000000A6     D
PCB$L_ASTQFL                    00000010     D          UCB$B_ERTCNT                    00000070     D
PCB$L_EFC2P                     00000058     D          UCB$B_ERTMAX                    00000071     D
PCB$L_EFC3P                     0000005C     D          UCB$B_FEX                       00000076     D
PCB$L_EFCS                      00000050     D          UCB$B_FIPL                      0000000B     D
PCB$L_EFCU                      00000054     D          UCB$B_LOCSRV                    0000003C     D
PCB$L_EFWM                      00000040     D          UCB$B_OFFNDX                    00000094     D
PCB$L_JIB                       00000078     D          UCB$B_OFFRTC                    00000095     D
PCB$L_OWNER                     0000001C     D          UCB$B_REMSRV                    0000003D     D
PCB$L_PHD                       00000064     D          UCB$B_SECTORS                   0000003C     D
PCB$L_PHYPCB                    00000018     D          UCB$B_SLAVE                     00000074     D
PCB$L_PID                       00000060     D          UCB$B_SPR                       00000075     D
PCB$L_PQB                       0000004C     D          UCB$B_STATE                     00000052     D
PCB$L_SQBL                      00000004     D          UCB$B_TRACKS                    0000003D     D
PCB$L_SQFL                      00000000     D          UCB$B_TT_CRFILL                 0000009D     D
PCB$L_STS                       00000024     D          UCB$B_TT_DECRF                  000000A1     D
PCB$L_UIC                       00000088     D          UCB$B_TT_DELFF                  000000A2     D
PCB$L_WSSWP                     00000020     D          UCB$B_TT_DESPEE                 000000A0     D
PCB$L_WTIME                     00000028     D          UCB$B_TT_DETYPE                 000000A4     D
PCB$Q_PRIV                      0000007C     D          UCB$B_TT_LFFILL                 0000009E     D
PCB$T_LNAME                     00000068     D          UCB$B_TT_SPEED                  0000009C     D
PCB$T_TERMINAL                  00000044     D          UCB$B_TYPE                      0000000A     D
PCB$W_APTCNT                    00000030     D          UCB$B_VERTSZ                    0000003F     D
PCB$W_ASTCNT                    00000038     D          UCB$C_LENGTH                    00000074     D
PCB$W_BIOCNT                    0000003A     D          UCB$C_MB_LENGTH                 00000090     D
PCB$W_BIOLM                     0000003C     D          UCB$C_TT_LENGTH                 000000BC     D
PCB$W_DIOCNT                    0000003E     D          UCB$K_LENGTH                    00000074     D
PCB$W_DIOLM                     00000040     D          UCB$K_MB_LENGTH                 00000090     D
PCB$W_GPGCNT                    00000034     D          UCB$K_TT_LENGTH                 000000BC     D
PCB$W_GRP                       0000008A     D          UCB$L_AMB                       00000054     D
PCB$W_MEM                       00000088     D          UCB$L_ASTQBL                    00000010     D
PCB$W_MTXCNT                    0000000E     D          UCB$L_ASTQFL                    0000000C     D
PCB$W_PPGCNT                    00000036     D          UCB$L_CPID                      0000005C     D
PCB$W_PRCCNT                    00000042     D          UCB$L_CRB                       00000020     D
PCB$W_SIZE                      00000008     D          UCB$L_DDB                       00000024     D
PCB$W_STATE                     0000002C     D          UCB$L_DEVCHAR                   00000034     D
PCB$W_TMBU                      00000032     D          UCB$L_DEVDEPEND                 0000003C     D
PR$_IPL                       = 00000012     D          UCB$L_DPC                       00000080     D
PR$_SIRR                      = 00000014     D          UCB$L_DUETIM                    0000005C     D
PRI$_IOCOM                    = 00000001     D          UCB$L_DX_BFPNT                  0000009C     D
PRI$_TICOM                    = 00000004     D          UCB$L_DX_BUF                    00000098     D
PRI$_TOCOM                    = 00000003     D          UCB$L_DX_RXDB                   000000A0     D
PRITBL                          00000000 R   D  03      UCB$L_EMB                       00000078     D
QNXTSEG                         00000119 R   D  03      UCB$L_FIRST                     00000014     D
SCH$GL_PCBVEC                   ********   X   03      UCB$L_FPC                       0000000C     D
```

UCB$L_FQBL	00000004	D	UCB$W_SRCADDR		0000001A	D			
UCB$L_FQFL	00000000	D	UCB$W_STS		00000058	D			
UCB$L_FR3	00000010	D	UCB$W_TT_DESIZE		000000A5	D			
UCB$L_FR4	00000014	D	UCB$W_UNIT		00000048	D			
UCB$L_IOQBL	00000044	D	UCB$W_VPROT		0000001A	D			
UCB$L_IOQFL	00000040	D	UNLK		000003B0	R	D	03	
UCB$L_IRP	0000004C	D	UNLOCK		0000006A	R	D	03	
UCB$L_LINK	0000002C	D	UNLOCK_MORE		00000381	R	D	03	
UCB$L_LOGADR	00000064	D	VA$M_BYTE	=	000001FF		D		
UCB$L_MAXBLOCK	00000084	D	VA$S_BYTE	=	00000009		D		
UCB$L_MB_MBX	0000007C	D	VCB$B_BLOCKFACT		00000052		D		
UCB$L_MB_PORT	0000008C	D	VCB$B_CUR_RVN		0000002F		D		
UCB$L_MB_RAST	00000078	D	VCB$B_EOFDELTA		0000004E		D		
UCB$L_MB_SHB	00000080	D	VCB$B_IBMAPSIZE		00000038		D		
UCB$L_MB_WAST	00000074	D	VCB$B_IBMAPVBN		0000003A		D		
UCB$L_MB_WIOQBL	00000088	D	VCB$B_LRU_LIM		00000049		D		
UCB$L_MB_WIOQFL	00000084	D	VCB$B_QNAMECNT		0000000B		D		
UCB$L_MEDIA	0000008C	D	VCB$B_RESFILES		0000004F		D		
UCB$L_NT_DATSSB	00000074	D	VCB$B_SBMAPSIZE		00000039		D		
UCB$L_NT_INTSSB	00000078	D	VCB$B_SBMAPVBN		0000003B		D		
UCB$L_OPCNT	00000060	D	VCB$B_STATUS		0000000B		D		
UCB$L_OWNUIC	0000001C	D	VCB$B_STATUS2		00000053		D		
UCB$L_PID	00000028	D	VCB$B_TM		0000002E		D		
UCB$L_RQBL	00000004	D	VCB$B_TYPE		0000000A		D		
UCB$L_RQFL	00000000	D	VCB$B_WINDOW		00000048		D		
UCB$L_SVAPTE	00000068	D	VCB$C_COMLEN		00000024		D		
UCB$L_SVPN	00000064	D	VCB$C_LENGTH		00000064		D		
UCB$L_TT_DECHAR	000000A8	D	VCB$C_MRKLEN		0000000B		D		
UCB$L_TT_RDUE	0000008C	D	VCB$K_COMLEN		00000024		D		
UCB$L_TT_RTIMOU	000000B8	D	VCB$K_LENGTH		00000064		D		
UCB$L_VCB	00000030	D	VCB$K_MRKLEN		0000000B		D		
UCB$T_PARTNER	000000CC	D	VCB$L_AQB		00000010		D		
UCB$W_BCNT	0000006E	D	VCB$L_BLOCKBL		00000004		D		
UCB$W_BCR	00000096	D	VCB$L_BLOCKFL		00000000		D		
UCB$W_BOFF	0000006C	D	VCB$L_CACHE		00000058		D		
UCB$W_BUFQUO	00000018	D	VCB$L_CUR_FID		00000024		D		
UCB$W_BYTESTOGO	0000003E	D	VCB$L_FCBBL		00000004		D		
UCB$W_CHARGE	0000004A	D	VCB$L_FCBFL		00000000		D		
UCB$W_CYLINDERS	0000003E	D	VCB$L_FREE		00000040		D		
UCB$W_DA	0000008C	D	VCB$L_HOME2LBN		00000028		D		
UCB$W_DC	0000008E	D	VCB$L_HOMELBN		00000024		D		
UCB$W_DEVBUFSIZ	0000003A	D	VCB$L_IBMAPLBN		00000030		D		
UCB$W_DEVSTS	0000005A	D	VCB$L_IXHDR2LBN		0000002C		D		
UCB$W_DIRSEQ	00000088	D	VCB$L_MAXFILES		00000044		D		
UCB$W_DSTADDR	00000018	D	VCB$L_MVL		00000034		D		
UCB$W_DX_BCR	000000A4	D	VCB$L_QUOCACHE		0000005C		D		
UCB$W_ECT	00000090	D	VCB$L_QUOTAFCB		00000054		D		
UCB$W_EC2	00000092	D	VCB$L_RVT		00000020		D		
UCB$W_ERRCNT	00000072	D	VCB$L_SBMAPLBN		00000034		D		
UCB$W_FUNC	0000007E	D	VCB$L_START_FID		00000028		D		
UCB$W_MB_SEED	00000000	D	VCB$L_ST_RECORD		00000030		D		
UCB$W_MSGCNT	00000016	D	VCB$L_USRLBLAST		00000044		D		
UCB$W_MSGMAX	00000014	D	VCB$L_VPBL		00000040		D		
UCB$W_NT_CHAN	0000007C	D	VCB$L_VPFL		0000003C		D		
UCB$W_OFFSET	0000008A	D	VCB$L_WCB		00000038		D		
UCB$W_REFC	00000050	D	VCB$T_QNAME		0000000C		D		
UCB$W_SIZE	00000008	D	VCB$T_VOLNAME		00000014		D		

```
VCB$W_CLUSTER                   0000003C     D
VCB$W_CUR_NUM                   00000024     D
VCB$W_CUR_SEQ                   00000026     D
VCB$W_EXTEND                    0000003E     D
VCB$W_FILEPROT                  0000004A     D
VCB$W_MCOUNT                    0000004C     D
VCB$W_MODE                      0000002C     D
VCB$W_PENDERR                   00000062     D
VCB$W_QUOSIZE                   00000060     D
VCB$W_RECORDSZ                  00000050     D
VCB$W_RVN                       0000000E     D
VCB$W_SIZE                      00000008     D
VCB$W_START_NUM                 00000028     D
VCB$W_START_SEQ                 0000002A     D
VCB$W_TRANS                     0000C00C     D
VIRTUAL                         000000E8 R   D  03
WCB$B_ACCESS                    0000000B     D
WCB$B_TYPE                      0000000A     D
WCB$C_LENGTH                    00000024     D
WCB$C_MAP                       00000024     D
WCB$K_LENGTH                    00000024     D
WCB$K_MAP                       00000024     D
WCB$L_FCB                       00000018     D
WCB$L_LBN                       00000002     D
WCB$L_ORGUCB                    00000010     D
WCB$L_P1_LBN                    00000026     D
WCB$L_P2_LBN                    0000002C     D
WCB$L_PID                       0000000C     D
WCB$L_PREVLBN                   FFFFFFFC     D
WCB$L_RVT                       0000001C     D
WCB$L_STVBN                     00000020     D
WCB$L_WLBL                      00000004     D
WCB$L_WLFL                      00000000     D
WCB$V_NOTFCP                =   00000002     D
WCB$W_ACON                      00000014     D
WCB$W_COUNT                     00000000     D
WCB$W_NMAP                      00000016     D
WCB$W_P1_COUNT                  00000024     D
WCB$W_P2_COUNT                  0000002A     D
WCB$W_PREVCOUNT                 FFFFFFFA     D
WCB$W_REFCNT                    0000000E     D
WCB$W_SIZE                      00000008     D
WQH$C_LENGTH                    0000000C     D
WQH$K_LENGTH                    0000000C     D
WQH$L_WQBL                      00000004     D
WQH$L_WQFL                      00000000     D
WQH$W_WQCNT                     00000008     D
WQH$W_WQSTATE                   0000000A     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	0C000011 (17.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	000003C2 (962.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	216.1 (3)	#-216.1 (3)
$MODULE	00000000-R	115.2 (2)	216.1 (3)
AACPERR	00000007-R	115.3 (2)	
ACB$B_RMOD	0000000B		#-288 (3)
ACB$L_KAST	00000018		#-275 (3) #-278 (3)
ACB$V_QUOTA	=00000006		#-831.36 (7)
ACCVIO	0000037C-R	831.77 (8)	#-701.63 (6) #-701.70 (6) #-831.71 (8)
BUFIO	00000096-R	260 (3)	#-175 (3)
BUFPOST	000001E8-R	701.13 (6)	275 (3)
BUG_CHECK	00000000-XR		639.13 (5) 639.5 (5)
CCB$L_DIRP	0000000C		#-831.18 (7) #-831.20 (7)
CCB$L_UCB	00000000		#-831.22 (7)
CCB$W_IOC	0000000A		#-831.16 (7) #-831.21 (7)
CTL$GL_CCBBASE	00000000-XR		831.15 (7)
CXB$L_LINK	=00000010		#-701.60 (6) #-701.65 (6)
CXB$W_LENGTH	=0000000C		#-701.53 (6) #-701.55 (6) #-701.56 (6) #-701.58 (6)
DEV$V_SQD	00000000-XR		#-301.22 (4)
DIRIO	0000004E-R	188 (3)	
DIRPOST	000002BE-R	831.4 (7)	278 (3)
DS_ERRSUP	00000000-XR		216.1 (3)
EXE$DEANONPAGED	00000000-XR		#-277 (3) #-701.67 (6) #-701.72 (6) #-831.13 (7)
			#-831.40 (7) #-831.52 (7)
EXE$INSIOQ	00000000-XR		600.32 (5)
IOC$CVTLOGPHY	00000000-XR		600.31 (5)
IOC$DIRPOST1	00000302-R	831.29 (7)	
IOC$GL_PSBL	00000000-XR		600.41 (5)
IOC$GL_PSFL	00000000-XR		156.1 (3)
IOC$IOPOST	00000004-R	153 (3)	
IOC$MAPVBLK	00000000-XR		600.20 (5)
IOC$QNXTSEG	0000014C-R	600.3 (5)	#-301.31 (4)
IOC$QNXTSEG1	00000158-R	600.18 (5)	
IOC$QTOACP	000001A5-R	632.5 (5)	#-301.55 (4)
IOC$WAKACP	000001B6-R	639.3 (5)	#-831.23 (7)
IOPOST	0000000D-R	156.1 (3)	#-167 (3) #-290 (3) #-301.32 (4)
IOPOST_KAST	000000DA-R	287 (3)	
IPL$_IOPOST	=00000004		#-600.43 (5)
IPL$_SYNCH	=00000007		#-639.4 (5)
IRP$B_EFN	00000022		#-285 (3)
IRP$B_RMOD	0000000B		701.46 (6) 831.32 (7) 831.36 (7) 831.69 (8)
IRP$L_DIAGBUF	00000044		#-301.37 (4) #-301.53 (4) #-831.12 (7) #-831.7 (7)
IRP$L_EXTEND	0000004C		1028 (9) #-831.47 (7)
IRP$L_IOSB	00000024		#-831.30 (7)
IRP$L_IOST1	00000034		#-193 (3) #-194 (3) #-301.15 (4) #-301.19 (4)
			#-301.20 (4) #-600.39 (5) #-831.34 (7) #-831.77 (8)
IRP$L_IOST2	00000038		#-600.40 (5)
IRP$L_PID	00000010		#-169 (3) #-283 (3)
IRP$L_SEGVBN	00000040		#-301.18 (4) #-600.6 (5)
IRP$L_SVAPTE	0000002C		#-177 (3) #-273 (3) #-301.28 (4) #-301.53 (4)
			#-701.15 (6) #-701.64 (6) #-701.71 (6)
IRP$L_UCB	0000001C		#-301.21 (4) #-301.30 (4) #-632.10 (5)

```
IRP$L_WIND                      00000018                #-600.4  (5)    #-632.7  (5)
IRP$M_PAGIO                   =00000004                 #-181    (3)
IRP$M_SWAPIO                  =00000040                 #-181    (3)
IRP$M_VIRTUAL                 =00000010                 #-301.52 (4)
IRP$V_BUFIO                   =00000000                 #-174    (3)    #-279    (3)
IRP$V_CHAINED                 =00000005                 #-701.18 (6)
IRP$V_COMPLX                  =00000003                 #-701.17 (6)
IRP$V_DIAGBUF                 =00000007                 #-831.5  (7)
IRP$V_EXTEND                  =0000000B                 #-200    (3)    #-831.35 (7)
IRP$V_FILACP                  =0000000C                 #-261    (3)
IRP$V_FUNC                    =00000001                 #-276    (3)
IRP$V_PAGIO                   =00000002                  179     (3)
IRP$V_SWAPIO                  =00000006                  180     (3)
IRP$V_TERMIO                  =00000009                 #-280    (3)
IRP$V_VIRTUAL                 =00000004                 #-192    (3)
IRP$W_ABCNT                     0000003C                #-301.16 (4)    #-301.19 (4)    #-301.23 (4)
IRP$W_BCNT                      00000032                #-190    (3)    #-301.24 (4)    #-600.21 (5)    #-600.24 (5)
                                                        #-600.5  (5)    #-632.6  (5)    #-701.16 (6)
IRP$W_BOFF                      00000030                #-191    (3)
IRP$W_CHAN                      00000028                #-831.14 (7)
IRP$W_OBCNT                     0000003E                #-194    (3)    #-301.23 (4)    #-301.36 (4)    #-301.54 (4)
IRP$W_SIZE                      00000008                #-831.8  (7)
IRP$W_STS                       0000002A                 174     (3)    #-181    (3)     192     (3)     200     (3)
                                                         261     (3)     276     (3)     279     (3)     280     (3)
                                                        #-301.52 (4)     701.17 (6)     701.18 (6)     831.35 (7)
                                                         831.5   (7)
IRPE$L_BCNT1                    00000034                #-1039   (9)
IRPE$L_BCNT2                    00000040                #-1045   (9)
IRPE$L_EXTEND                   0000004C                 1028    (9)    #-1035   (9)    #-831.51 (7)
IRPE$L_SVAPTE1                  00000020                #-1036   (9)
IRPE$L_SVAPTE2                  00000038                #-1042   (9)
IRPE$V_EXTEND                 =0000000B                 #-1048   (9)    #-831.50 (7)
IRPE$W_BOFF1                    00000030                #-1038   (9)
IRPE$W_BOFF2                    0000003C                #-1044   (9)
IRPE$W_STS                      0000002A                 1048    (9)     831.50 (7)
MMG$UNLOCK                      00000000-XR              1061.1  (9)     198.1  (3)
MOVBUF                          00000349-R    831.62 (8) #-701.38 (6)    #-831.10 (7)
NOTACP                          000000A1-R    263    (3) #-261    (3)
NOTFCPWCB                       000001D2-R    639.12 (5) #-632.9  (5)
PAGIO_OR_SWAPIO                 00000085-R    216    (3) #-182    (3)
PCB$W_BIOCNT                    0000003A                 #-260    (3)
PCB$W_DIOCNT                    0000003E                 #-176    (3)    #-262    (3)    #-301.51 (4)    #-600.19 (5)
PR$_IPL                       =00000012                  #-639.4  (5)    #-639.6  (5)
PR$_SIRR                      =00000014                  #-600.43 (5)
PRI$_IOCOM                    =00000001                   118     (2)     120     (2)
PRI$_TICOM                    =00000004                   121     (2)
PRI$_TOCOM                    =00000003                   119     (2)
PRITBL                          00000000-R    117    (2) #-284    (3)
QNXTSEG                         00000119-R    301.27 (4)
SCH$GL_PCBVEC                   00000000-XR              #-172.1  (3)
SCH$POSTEF                      00000000-XR              #-286    (3)
SCH$QAST                        00000000-XR               288.1  (3)     831.38 (7)
SS$_ACCVIO                      00000000-XR              #-831.77 (8)
SS$_ILLBLKNUM                   00000000-XR              #-600.39 (5)
UCB$L_DEVCHAR                   00000034                  301.22 (4)
UCB$L_MAXBLOCK                  00000084                 #-600.29 (5)
UNLK                            000003B0-R    1060   (9) #-1040   (9)    #-1046   (9)
```

```
UNLOCK                          0000006A-R   197    (3)     #-192     (3)     #-301.38 (4)
UNLOCK_MORE                     00000381-R   1030   (9)     #-201     (3)
VA$M_BYTE                       =000001FF                   #-701.24  (6)     #-701.44 (6)     #-831.67 (8)
VA$S_BYTE                       =00000009                   #-1061    (9)     #-198    (3)     #-301.17 (4)     #-301.26 (4)
                                                            #-600.26  (5)
VIRTUAL                         000000E3-R   301.14 (4)     #-164     (3)
WCB$B_ACCESS                    00000008                      632.8   (5)
WCB$V_NOTFCP                    =00000002                   #-632.8   (5)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	91 (2)	91 (2)
$AQBDEF	2	92 (2)	92 (2)
$CADEF	1	93 (2)	93 (2)
$CCBDEF	1	94 (2)	94 (2)
$CXBDEF	2	95 (2)	95 (2)
$DEFINI	1	91 (2)	100 (2) 101 (2) 102 (2) 103 (2) 104 (2)
			105 (2) 106 (2) 107 (2) 108 (2) 109 (2)
			110 (2) 111 (2) 91 (2) 92 (2) 93 (2)
			94 (2) 95 (2) 96 (2) 97 (2) 98 (2)
			99 (2)
$IPLDEF	1	96 (2)	96 (2)
$IRPDEF	4	97 (2)	97 (2)
$IRPEDEF	2	98 (2)	98 (2)
$JIBDEF	3	99 (2)	99 (2)
$PCBDEF	4	100 (2)	100 (2)
$PFNDEF	2	101 (2)	101 (2)
$PHDDEF	6	102 (2)	102 (2)
$PRDEF	4	103 (2)	103 (2)
$PRIDEF	1	104 (2)	104 (2)
$PTEDEF	3	105 (2)	105 (2)
$PUSHADR	1	216.1 (3)	216.1 (3)
$RSNDEF	1	106 (2)	106 (2)
$UCBDEF	10	107 (2)	107 (2)
$VADEF	1	108 (2)	108 (2)
$VCBDEF	6	109 (2)	109 (2)
$WCBDEF	3	110 (2)	110 (2)
$WQHDEF	1	111 (2)	111 (2)
ASSUME	1	179 (3)	1028 (9) 179 (3) 180 (3)
BUG_CHECK	1	639.5 (5)	639.13 (5) 639.5 (5)
DSBINT	1	639.4 (5)	639.4 (5)
ENBINT	1	639.6 (5)	639.6 (5)
ERRSUP_S	1	216.1 (3)	216.1 (3)
IFNOWRT	1	701.28 (6)	701.28 (6) 701.48 (6) 831.33 (7) 831.71 (8)
MODNAM	1	115.2 (2)	115.2 (2)
SOFTINT	1	600.43 (5)	600.43 (5)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.17	00:00:00.30
Command processing	148	00:00:00.76	00:00:02.47
Pass 1	720	00:00:20.27	00:00:32.86
Symbol table sort	0	00:00:01.48	00:00:01.75
Pass 2	177	00:00:04.75	00:00:09.55
Symbol table output	48	00:00:00.32	00:00:00.84
Psect synopsis output	7	00:00:00.02	00:00:00.02
Cross-reference output	53	00:00:00.55	00:00:00.59

```
Assembler run totals          1192     00:00:28.31    00:00:48.39

The working set limit was 1000 pages.
80436 bytes (158 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 988 non-local and 44 local symbols.
678 source lines were read in Pass 1, producing 0 object records in Pass 2.
124 pages of virtual memory were used to define 39 macros.
```

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]D3.MLB:218	3
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	10
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	14
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	35

```
1482 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) IOPOST/UPDA=(IOPOST.UPD,IOPOST.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 71 CANCEL I/O ON CHANNEL
(1) 175 FILL DIAGNOSTIC BUFFER
(1) 210 RELEASE I/O CHANNEL
(1) 265 REQUEST I/O CHANNEL
(1) 323 I/O REQUEST COMPLETION PROCESSING
(1) 372 INITIATE I/O FUNCTION ON DEVICE
(1) 414 RELEASE BUFFERED DATAPATH
(1) 462 REQUEST BUFFERED DATAPATH
(1) 516 RELEASE UNIBUS MAP REGISTERS
(1) 569 REQUEST UNIBUS MAP REGISTERS
(1) 604 ALTER UBA MAP REGISTER BITMAP
(1) 637 SEARCH MAP REGISTER BITMAP AND ALLOCATE MAP REGISTERS
(1) 690 RETURN TO CALLER
(1) 709 WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
(1) 743 WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
(1) 779 ALLOCATE SYSTEM PAGE TABLE

```
02                      0000     .1          .TITLE  IOSNPG  *** IOSNPG services for QIO
02                      0000     .2          .IDENT  /06-03/
-2                      0000     3  ;*****************************************************************************
                        0000     4  ;*                                                                           *
                        0000     5  ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000     6  ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
                        0000     7  ;*                                                                           *
                        0000     8  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED    *
                        0000     9  ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
                        0000    10  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
                        0000    11  ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000    12  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
                        0000    13  ;*  TRANSFERRED.                                                             *
                        0000    14  ;*                                                                           *
                        0000    15  ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
                        0000    16  ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                        0000    17  ;*  CORPORATION.                                                             *
                        0000    18  ;*                                                                           *
                        0000    19  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS    *
                        0000    20  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000    21  ;*                                                                           *
                        0000    22  ;*****************************************************************************
                        0000    23  ;
                        0000    24  ; D. N. CUTLER 13-JUN-76
                        0000    25  ;
                        0000    26  ;
                        0000    27  ; NONPAGED I/O RELATED SUBROUTINES
                        0000    28  ;
                        0000    29  ; MODIFICATION HISTORY:
                        0000    30  ;
02                      0000    .1  ;
02                      0000    .2  ;       CHANGE PARAMETERS TO WAIT FOR INTERRUPTS.
02                      0000    .3  ;
02                      0000    .4  ;               Dave Butenhof   13-may-1980, Version 5.4
02                      0000    .5  ;       01      Modify VMS V2.0 sources for Supervisor environment
02                      0000    .6  ;               Roger Riggs, 23-July-1980, Version 5.5
02                      0000    .7  ;       02      Changed word offsets to longword
03                      0000    .8  ;               Dave Butenhof   24-nov-1980, Version 6.2
03                      0000    .9  ;       03      Restore entry point IOC$ALLOSPT to satisfy device drivers
03                      0000   .10  ;               using it.  It remains essentially "dummied out," since
03                      0000   .11  ;               Supervisor I/O drivers do not require system space
                        0000    31  ;       0206    KDM0086         KATHLEEN D. MORSE       07-JAN-1980
                        0000    32  ;               CHANGE UCB$W_DEVSTS TO UCB$W_STS, AND EMB$W_DV_DEVSTS TO
                        0000    33  ;               EMB$W_DV_STS.
                        0000    34  ;
                        0000    35  ;       0205    RIP0033         R. HUSTVEDT             16-OCT-1979
                        0000    36  ;               CHANGE PCB$W_BYTLM TO JIB$L_BYTLM.
                        0000    37  ;
                        0000    38  ;       0204    NPK0002         N. KRONENBERG           21-SEP-1979
                        0000    39  ;               MODIFIED IOC$RELDATAP TO ZERO THE DATAPATH
                        0000    40  ;               NUMBER IN THE CALLERS CRB.
                        0000    41  ;
                        0000    42  ;       0203    NPKCOMET        N. KRONENBERG           11-FEB-1979
                        0000    43  ;               MODIFIED IOC$REQDATAP AND IOC$RELDATAP TO HANDLE
                        0000    44  ;               6 BIT DATAPATH NUMBERS.
                        0000    45  ;
                        0000    46  ;       0202    LMK0001         LEN KAWELL              07-FEB-1979
```

```
                        0000    47 ;
                        0000    48 ;
```

ADDED IOC$ALLOSPT ROUTINE THAT ALLOCATES SYSTEM
PAGE TABLE ENTRIES.

```
                        0000     50 ;
                        0000     51 ;
                        0000     52 ; MACRO LIBRARY CALLS
                        0000     53 ;
                        0000     54
                        0000     55          $ADPDEF                         ;DEFINE ADP OFFSETS
                        0000     56          $CADEF                          ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS
                        0000     57          $CRBDEF                         ;DEFINE CRB OFFSETS
                        0000     58          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     59          $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000     60          $EMBDEF                         ;DEFINE EMB OFFSETS
                        0000     61          $IDBDEF                         ;DEFINE IDB OFFSETS
                        0000     62          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000     63          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000     64          $JIBDEF                         ;DEFINE JIB OFFSETS
                        0000     65          $LOGDEF                         ;DEFINE LOGICAL NAME BLOCK OFFSETS
                        0000     66          $PCBDEF                         ;DEFINE PCB OFFSETS
                        0000     67          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     68          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000     69          $VECDEF                         ;DEFINE CRB VECTOR OFFSETS
```

```
                                0000     71          .SBTTL  CANCEL I/O ON CHANNEL
                                0000     72 ;+
                                0000     73 ; IOC$CANCELIO - CANCEL I/O ON CHANNEL
                                0000     74 ;
                                0000     75 ; THIS ROUTINE IS A DEVICE INDEPENDENT CANCEL I/O ROUTINE THAT CONDITIONALLY
                                0000     76 ; MARKS THE UCB SUCH THAT THE CURRENT I/O REQUEST WILL BE CANCELED IF CONDITIONS
                                0000     77 ; WARRANT SUCH A ACTION.
                                0000     78 ;
                                0000     79 ; INPUTS:
                                0000     80 ;
                                0000     81 ;       R2 = NEGATIVE OF THE CHANNEL NUMBER.
                                0000     82 ;       R3 = CURRENT IO PACKET.
                                0000     83 ;       R4 = PCB ADDRESS.
                                0000     84 ;       R5 = UCB ADDRESS.
                                0000     85 ;
                                0000     86 ; OUTPUTS:
                                0000     87 ;
                                0000     88 ;       IF THE DEVICE IS BUSY, THE REQUEST IS FOR THE CURRENT PROCESS, AND
                                0000     89 ;       THE I/O WAS ISSUED FROM THE DESIGNATED CHANNEL, THEN THE CANCEL I/O
                                0000     90 ;       BIT IS SET IN THE CORRESPONDING UCB.
                                0000     91 ;
                                0000     92 ;       R2, R3, R4, AND R5 ARE PRESERVED ACROSS CALL.
                                0000     93 ;-
                                0000     94
01                         00000000      .1          .PSECT  SEP, SHR, EXE, WRT, LONG        ; **** SUPERVISOR ****
-1                              0000     96 IOC$CANCELIO::                           ;CANCEL I/O ON CHANNEL
                11 58 A5   08   E1  0000  97          BBC     #UCB$V_BSY,UCB$W_STS(R5),10$ ;IF CLR, DEVICE NOT BUSY
                60 A4    0C A3  D1  0005  98          CMPL    ;RP$L_PID(R3),PCB$L_PID(R4) ;PROCESS ID MATCH?
                               0A  12  000A  99          BNEQ    10$                         ;IF NEQ NO
                   28 A3       52  B1  000C 100          CMPW    R2,IRP$W_CHAN(R3)           ;CHANNEL NUMBER MATCH
                               04  12  0010 101          BNEQ    10$                         ;IF NEQ NO
                   58 A5       08  A8  0012 102          BISW    #UCB$M_CANCEL,UCB$W_STS(R5) ;SET CANCEL PENDING
                               05  0016 103 10$:    RSB                                     ;
```

```
                                0017   175          .SBTTL  FILL DIAGNOSTIC BUFFER
                                0017   176 ;+
                                0017   177 ; IOC$DIAGBUFILL - FILL DIAGNOSTIC BUFFER
                                0017   178 ;
                                0017   179 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION, BUT BEFORE RELEASING
                                0017   180 ; THE I/O CHANNEL, TO FILL THE FINAL DEVICE PARAMETERS INTO AN INTERNAL DIAG-
                                0017   181 ; NOSTIC BUFFER IF ONE IS SPECIFIED.
                                0017   182 ;
                                0017   183 ; INPUTS:
                                0017   184 ;
                                0017   185 ;       R4 = ADDRESS OF DEVICE CSR REGISTER.
                                0017   186 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0017   187 ;
                                0017   188 ; OUTPUTS:
                                C017   189 ;
                                0017   190 ;       IF A DIAGNOSTIC BUFFER WAS SPECIFIED IN THE ORIGINAL REQUEST, THEN
                                0017   191 ;       THE COMPLETION TIME, FINAL ERROR COUNTERS, AND DEVICE REGISTERS ARE
                                0017   192 ;       FILLED INTO THE DIAGNOSTIC BUFFER.
                                0017   193 ;-
                                0017   194
                                0017   195 IOC$DIAGBUFILL::                           ;FILL DIAGNOSTIC BUFFER
      53    4C A5    D0         0017   196          MOVL    UCB$L_IRP(R5),R3         ;GET ADDRESS OF I/O PACKET
  27 2A A3     07    E1         001B   197          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ;IF CLR, NO DIAGNOSTIC BUFFER
      50    44 B3    D0         0020   198          MOVL    @IRP$L_DIAGBUF(R3),R0    ;GET ADDRESS OF INTERNAL BUFFER DATA AREA
         50    08    C0         0024   199          ADDL    #8,R0                    ;POINT PAST START TIME
    00000000'EF    7D         0027   200          MOVQ    EXE$GQ_SYSTIME,(R0)+     ;INSERT COMPLETION TIME
                  80              002D
      80    70 A5    3C         002E   201          MOVZWL  UCB$B_ERTCNT(R5),(R0)+   ;INSERT FINAL ERROR COUNTERS
      52    24 A5    D0         0032   202          MOVL    UCB$L_DDB(R5),R2         ;GET ADDRESS OF DDB
      52    0C A2    D0         0036   203          MOVL    DDB$L_DDT(R2),R2         ;GET ADDRESS OF DDT
01    51    10 A2    D0         003A    .1          MOVL    DDT$L_REGDUMP(R2),R1     ;FETCH ADDRESS OF DUMP ROUTINE
01    03 51    10    E0         003E    .2          BBS     #16,R1,5$                ;+++ BRANCH IF SUPERVISOR ABSOLUTE
01       51    52    C0         0042    .3          ADDL    R2,R1                    ;+++ INCLUDE BASE ADDRESS
-3             61    16         0045   207 5$:      JSB     (R1)                     ;CALL DEVICE SPECIFIC REGISTER DUMP ROUTINE
                     05         0047   208 10$:     RSB                              ;
```

```
                                0048   210          .SBTTL  RELEASE I/O CHANNEL
                                0048   211 ;+
                                0048   212 ; IOC$RELCHAN - RELEASE ALL I/O CHANNELS
                                0048   213 ; IOC$RELSCHAN - RELEASE SECONDARY I/O CHANNEL
                                0048   214 ;
                                0048   215 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION TO RELEASE ALL
                                0048   216 ; CHANNELS THE I/O WAS BEING PERFORMED ON.
                                0048   217 ;
                                0048   218 ; INPUTS:
                                0048   219 ;
                                0048   220 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0048   221 ;
                                0048   222 ; OUTPUTS:
                                0048   223 ;
                                0048   224 ;       THE CHANNELS ARE RELEASED AND AN ATTEMPT IS MADE TO REMOVE THE NEXT
                                0048   225 ;       WAITING DRIVER PROCESS FROM EACH CHANNEL QUEUE. IF A DRIVER PROCESS
                                0048   226 ;       IS WAITING, THEN THE CHANNEL IS ASSIGNED TO THAT DRIVER PROCESS AND
                                0048   227 ;       IT IS CALLED VIA A JSB TO ITS CHANNEL WAIT RETURN ADDRESS. WHEN THE
                                0048   228 ;       CALLED DRIVER PROCESS RETURNS, A RETURN IS MADE TO THE DRIVER PROCESS
                                0048   229 ;       THAT RELEASED THE CHANNEL. IF THERE IS NO DRIVER PROCESS WAITING FOR
                                0048   230 ;       THE CHANNEL, THEN THE CHANNEL STATUS IS SET TO IDLE.
                                0048   231 ;
                                0048   232 ;       R3 AND R4 ARE PRESERVED ACROSS CALL.
                                0048   233 ;-
                                0048   234
                                0048   235          .ENABL  LSB
                                0048   236 IOC$RELSCHAN::                              ;RELEASE SECONDARY I/O CHANNEL
 50   20 A5   D0                0048   237          MOVL    UCB$L_CRB(R5),R0           ;GET ADDRESS OF PRIMARY CRB
 50   10 A0   D0                004C   238          MOVL    CRB$L_LINK(R0),R0          ;GET ADDRESS OF SECONARY CRB
        10   11                 0050   239          BRB     20$                        ;
                                0052   240 IOC$RELCHAN::                               ;RELEASE I/O CHANNEL
 50   20 A5   D0                0052   241          MOVL    UCB$L_CRB(R5),R0           ;GET ADDRESS OF PRIMARY CRB
 50   10 A0   D0                0056   242          MOVL    CRB$L_LINK(R0),R0          ;GET ADDRESS OF SECONDARY CRB
        02   13                 005A   243          BEQL    10$                        ;IF EQL NONE
        04   10                 005C   244          BSBB    20$                        ;RELEASE SECONDARY CHANNEL
 50   20 A5   D0                005E   245 10$:     MOVL    UCB$L_CRB(R5),R0           ;GET ADDRESS OF PRIMARY CRB
25 0E A0   00   E1              0062   246 20$:     BBC     #CRB$V_BSY,CRB$B_MASK(R0),30$ ;IF CLR, THEN CHANNEL NOT BUSY
 51   1C A0   D0                0067   247          MOVL    CRB$L_INTD+VEC$L_IDB(R0),R1 ;GET ADDRESS OF IDB
 04 A1   55   D1                006B   248          CMPL    R5,IDB$L_OWNER(R1)         ;DRIVER PROCESS OWN CHANNEL?
        1B   12                 006F   249          BNEQ    30$                        ;IF NEQ NO
 52   00 B0   0F                0071   250          REMQUE  @CRB$L_WQFL(R0),R2         ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
        16   1D                 0075   251          BVS     40$                        ;IF VS NO DRIVER PROCESS WAITING
        38   BB                 0077   252          PUSHR   #^M<R3,R4,R5>              ;SAVE CONTEXT OF CURRENT DRIVER PROCESS
    55  52   D0                0079   253          MOVL    R2,R5                      ;COPY ADDRESS OF DRIVER PROCESS FORK BLOCK
 53   10 A5   D0                007C   254          MOVL    UCB$L_FR3(R5),R3           ;LOAD WAITING DRIVER PROCESS CONTEXT
    54  61   D0                0080   255          MOVL    IDB$L_CSR(R1),R4           ;SET ASSIGNED CHANNEL CSR ADDRESS
 04 A1   55   D0                0083   256          MOVL    R5,IDB$L_OWNER(R1)         ;SET ADDRESS OF OWNER PROCESS UCB
      0C B5   16                0087   257          JSB     @UCB$L_FPC(R5)             ;CALL DRIVER AT CHANNEL WAIT RETURN ADDRESS
        38   BA                 008A   258          POPR    #^M<R3,R4,R5>              ;RESTORE PREVIOUS DRIVER PROCESS CONTEXT
        05   008C               259 30$:     RSB                                ;
    04 A1   D4                  008D   260 40$:     CLRL    IDB$L_OWNER(R1)            ;CLEAR OWNER UNIT UCB ADDRESS
 0E A0   01   8A                0090   261          BICB    #CRB$M_BSY,CRB$B_MASK(R0)  ;CLEAR CHANNEL BUSY
        05   0094               262          RSB                                ;
                                0095   263          .DSABL  LSB
```

```
                     0095   265          .SBTTL  REQUEST I/O CHANNEL
                     0095   266 ;+
                     0095   267 ; IOC$REQPCHANH - REQUEST PRIMARY I/O CHANNEL HIGH PRIORITY
                     0095   268 ; IOC$REQSCHANH - REQUEST SECONDARY I/O CHANNEL HIGH PRIORITY
                     0095   269 ; IOC$REQPCHANL - REQUEST PRIMARY I/O CHANNEL LOW PRIORITY
                     0095   270 ; IOC$REQSCHANL - REQUEST SECONDARY I/O CHANNEL LOW PRIORITY
                     0095   271 ;
                     0095   272 ; THESE ROUTINES ARE CALLED TO REQUEST AN I/O CHANNEL TO PERFORM AN I/O
                     0095   273 ; OPERATION ON.
                     0095   274 ;
                     0095   275 ; INPUTS:
                     0095   276 ;
                     0095   277 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     0095   278 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                     0095   279 ;
                     0095   280 ; OUTPUTS:
                     0095   281 ;
                     0095   282 ;       IF THE SPECIFIED I/O CHANNEL IS IDLE, THEN IT IS IMMEDIATELY
                     0095   283 ;       ASSIGNED TO THE CURRENT DRIVER PROCESS. ELSE THE DRIVER PROCESS
                     0095   284 ;       CONTEXT IS SAVED IN ITS FORK BLOCK, THE FORK BLOCK IS INSERTED
                     0095   285 ;       IN THE CHANNEL WAIT QUEUE, AND A RETURN TO THE DRIVER PROCESS'
                     0095   286 ;       CALLER IS EXECUTED.
                     0095   287 ;
                     0095   288 ;       WHEN THE CHANNEL IS ASSIGNED, THE CSR ADDRESS OF THE ASSIGNED
                     0095   289 ;       CONTROLLER IS RETURNED TO THE CALLER IN REGISTER R4.
                     0095   290 ;
                     0095   291 ;       R3 IS PRESERVED ACROSS CALL.
                     0095   292 ;-
                     0095   293
                     0095   294          .ENABL  LSB
                     0095   295 IOC$REQSCHANH::                          ;REQUEST SECONDARY I/O CHANNEL HIGH PRIORITY
 50   20 A5   D0     0095   296          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 50   10 A0   D0     0099   297          MOVL    CRB$L_LINK(R0),R0        ;GET ADDRESS OF SECONDARY CRB
           0E   11   009D   298          BRB     10$                      ;
                     009F   299 IOC$REQSCHANL::                          ;REQUEST SECONDARY I/O CHANNEL LOW PRIORITY
 50   20 A5   D0     009F   300          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 50   10 A0   D0     00A3   301          MOVL    CRB$L_LINK(R0),R0        ;GET ADDRESS OF SECONDARY CRB
           0D   11   00A7   302          BRB     20$                      ;
                     00A9   303 IOC$REQPCHANH::                          ;REQUEST PRIMARY I/O CHANNEL HIGH PRIORITY
 50   20 A5   D0     00A9   304          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 52   50      D0     00AD   305 10$:     MOVL    R0,R2                    ;SET ADDDRESS OF WAIT QUEUE LISTHEAD
           08   11   00B0   306          BRB     30$                      ;
                     00B2   307 IOC$REQPCHANL::                          ;REQUEST PRIMARY I/O CHANNEL LOW PRIORITY
 50   20 A5   D0     00B2   308          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 52   04 A0   D0     00B6   309 20$:     MOVL    CRB$L_WQBL(R0),R2        ;GET ADDRESS OF LAST ENTRY IN QUEUE
 51   1C A0   D0     00BA   310 30$:     MOVL    CRB$L_INTD+VEC$L_IDB(R0),R1 ;GET ADDRESS OF IDB
08 0E A0     00   E2 00BE   311          BBSS    #CRB$V_3SY,CRB$B_MASK(R0),40$ ;IF SET, THEN CHANNEL BUSY
 54   61      D0     00C3   312          MOVL    IDB$L_CSR(R1),R4         ;SET ASSIGNED CHANNEL CSR ADDRESS
 04 A1   55   D0     00C6   313          MOVL    R5,IDB$L_OWNER(R1)       ;SET OWNER UCB ADDRESS
           05   00   00CA   314          RSB                              ;
 10 A5   53   D0     00CB   315 40$:     MOVL    R3,UCB$L_FR3(R5)         ;SAVE R3 IN FORK BLOCK
    0C A5 8ED0     00CF   316          POPL    UCB$L_FPC(R5)            ;SAVE CHANNEL WAIT RETURN ADDRESS
 62   65   0E   00D3   317          INSQUE  UCB$L_FQFL(R5),CRB$L_WQFL(R2) ;INSERT DRIVER PROCESS IN CHANNEL WAIT
 04 A1   55   D1     00D6   318          CMPL    R5,IDB$L_OWNER(R1)       ;CURRENT DRIVER PROCESS OWNER?
           3E   13   00DA   319          BEQL    RELEASE                  ;IF EQL YES - RELEASE CHANNELS
                05   00DC   320          RSB                              ;
                     00DD   321          .DSABL  LSB
```

```
                                00DD    323          .SBTTL  I/O REQUEST COMPLETION PROCESSING
                                00DD    324 ;+
                                00DD    325 ; IOC$REQCOM - I/O REQUEST COMPLETE
                                00DD    326 ;
                                00DD    327 ; THIS ROUTINE IS ENTERED WHEN AN I/O OPERATION IS COMPLETED ON A
                                00DD    328 ; DEVICE UNIT. THE FINAL I/O STATUS IS STORED IN THE ASSOCIATED I/O
                                00DD    329 ; PACKET AND THE PACKET IS INSERTED IN THE I/O FINISH QUEUE FOR
                                00DD    330 ; I/O POST PROCESSING. DEVICE UNIT BUSY IS CLEARED AND AN ATTEMPT
                                00DD    331 ; IS MADE TO START ANOTHER I/O REQUEST ON THE DEVICE UNIT.
                                00DD    332 ;
                                00DD    333 ; INPUTS:
                                00DD    334 ;
                                00DD    335 ;       R0 = FIRST LONGWORD OF I/O STATUS.
                                00DD    336 ;       R1 = SECOND LONGWORD OF I/O STATUS.
                                00DD    337 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                00DD    338 ;
                                00DD    339 ; OUTPUTS:
                                00DD    340 ;
                                00DD    341 ;       THE I/O PACKET IS INSERTED IN THE I/O POST PROCESSING QUEUE
                                00DD    342 ;       AND DEVICE UNIT BUSY IS CLEARED. A SOFTWARE INTERRUPT IS
                                00DD    343 ;       REQUESTED TO INITIATE I/O POST PROCESSING.
                                00DD    344 ;-
                                00DD    345
                                00DD    346 IOC$REQCOM::                            ;I/O DONE PROCESSING
          53    4C A5    D0    00DD    347          MOVL    UCB$L_IRP(R5),R3       ;GET ADDRESS OF I/O PACKET
        34 A3     50    7D    00E1    348          MOVQ    R0,IRP$L_MEDIA(R3)     ;STORE FINAL I/O STATUS
              60 A5    D6    00E5    349          INCL    UCB$L_OPCNT(R5)        ;INCREMENT OPERATIONS COMPLETED
                               00E8    350
-5                             00E8    356
02                      63     0E    00E8     .1          INSQUE  (R3),@L^IOC$GL_PSBL    ;INSERT PACKET IN POST PROCESS QUEUE
                00000000'FF            00EA
-1                      03     12    00EF    358          BNEQ    10$                    ;IF NEQ NOT FIRST ENTRY IN QUEUE
                               00F1    359          SOFTINT #IPL$_IOPOST           ;INITIATE SOFTWARE INTERRUPT
       15 58 A5       02     E5    00F4    360 10$:     BBCC    #UCB$V_ERLOGIP,UCB$W_STS(R5),20$ ;IF CLR, ERROR LOG NOT IN PROGRESS
          52    78 A5    D0    00F9    361          MOVL    UCB$L_EMB(R5),R2       ;GET ADDRESS OF ERROR MESSAGE BUFFER
    1A A2     58 A5    B0    00FD    362          MOVW    UCB$W_STS(R5),EMB$W_DV_STS(R2) ;INSERT FINAL DEVICE STATUS
    10 A2     70 A5    B0    0102    363          MOVW    UCB$B_ERTCNT(R5),EMB$B_DV_ERTCNT(R2) ;INSERT FINAL ERROR COUNTERS
       12 A2     50    7D    0107    364          MOVQ    R0,EMB$Q_DV_IOSB(R2)   ;INSERT FINAL I/O STATUS
                FEF2'  30    010B    365          BSBW    ERL$RELEASEMB          ;RELEASE ERROR MESSAGE BUFFER
          53    40 B5    0F    010E    366 20$:     REMQUE  @UCB$L_IOQFL(R5),R3    ;REMOVE I/O PACKET FROM DEVICE UNIT QUEUE
                   09    1C    0112    367          BVC     IOC$INITIATE           ;IF VC INITIATE NEXT FUNCTION
             0100 8F     AA    0114    368          BICW    #UCB$M_BSY,UCB$W_STS(R5) ;CLEAR UNIT BUSY
                58 A5            0118
                               011A    369 RELEASE:                               ;RELEASE ALL CHANNELS
                  FF35    31    011A    370          BRW     IOC$RELCHAN            ;
```

```
                                011D   372          .SBTTL  INITIATE I/O FUNCTION ON DEVICE
                                011D   373 ;+
                                011D   374 ; IOC$INITIATE - INITIATE NEXT FUNCTION ON DEVICE
                                011D   375 ;
                                011D   376 ; THIS ROUTINE IS CALLED TO INITIATE THE NEXT FUNCTION ON A DEVICE BY CLEARING
                                011D   377 ; STATUS BITS, SETTING THE OPERATION START TIME IF A DIAGNOSTIC BUFFER IS
                                011D   378 ; SPECIFIED, AND CALLING THE DRIVER AT ITS START I/O ENTRY POINT.
                                011D   379 ;
                                011D   380 ; INPUTS:
                                011D   381 ;
                                011D   382 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                011D   383 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                011D   384 ;
                                011D   385 ; OUTPUTS:
                                011D   386 ;
                                011D   387 ;       CANCEL I/O, POWERFAIL, AND TIME OUT STATUS BITS ARE CLEARED, THE
                                011D   388 ;       CURRENT SYSTEM TIME IS FILLED INTO THE INTERNAL DIAGNOSTIC BUFFER
                                011D   389 ;       IF ONE IS SPECIFIED, AND THE DRIVER IS CALLED AT ITS START I/O ENTRY
                                011D   390 ;       POINT.
                                011D   391 ;-
                                011D   392
                                011D   393 IOC$INITIATE::                          ;INITIATE I/O FUNCTION
          4C A5    53   D0     011D   394          MOVL    R3,UCB$L_IRP(R5)       ;SAVE I/O PACKET ADDRESS
                                0121   395
-5                              0121   401
      68 A5    2C A3   7D     0121   402          MOVQ    IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5) ;COPY TRANSFER PARAMETERS
            0048 8F    AA     0126   403          BICW    #UCB$M_CANCEL!UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR CANCEL AND TIME OUT
               58 A5          012A
      0B 2A A3    07   E1     012C   404          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ;IF CLR, NO DIAGNOSTIC BUFFER
         50    44 B3   D0     0131   405          MOVL    @IRP$L_DIAGBUF(R3),R0  ;GET ADDRESS OF DIAGNOSTIC BUFFER DATA AREA
           00000000'EF   7D     0135   406          MOVQ    EXE$GQ_SYSTIME,(R0)    ;INSERT I/O OPERATION START TIME
                  60          0138
         50    24 A5   D0     013C   407 10$:      MOVL    UCB$L_DDB(R5),R0       ;GET ADDRESS OF DEVICE DATA BLOCK
         50    0C A0   D0     0140   408          MOVL    DDB$L_DDT(R0),R0       ;GET ADDRESS OF DRIVER DISPATCH TABLE
01          51    60   D0     0144    .1          MOVL    DDT$L_START(R0),R1     ;FETCH ADDRESS OF START ENTRY POINT
01       03 51    10   E0     0147    .2          BBS     #16,R1,20$             ;+++ BRANCH IF SUPERVISOR ABSOLUTE
01          51    50   C0     014B    .3          ADDL    R0,R1                  ;+++ CHANGE OFFSET TO ABSOLUTE
-3                61   17     014E   412 20$:      JMP     (R1)                   ;START I/O OPERATION
```

```
                        0150   414          .SBTTL  RELEASE BUFFERED DATAPATH
                        0150   415 ;+
                        0150   416 ; IOC$RELDATAP - RELEASE BUFFERED DATAPATH
                        0150   417 ;
                        0150   418 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION TO RELEASE A UNIBUS
                        0150   419 ; ADAPTER BUFFERED DATAPATH.
                        0150   420 ;
                        0150   421 ; INPUTS:
                        0150   422 ;
                        0150   423 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                        0150   424 ;
                        0150   425 ; OUTPUTS:
                        0150   426 ;
                        0150   427 ;       IF THE BUFFERED DATAPATH IS PERMANENTLY ASSIGNED TO THE ASSOCIATED
                        0150   428 ;       I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                        0150   429 ;       ELSE THE BUFFERED DATAPATH IS RELEASED AND AN ATTEMPT IS MADE TO
                        0150   430 ;       REMOVE THE NEXT WAITING DRIVER PROCESS FROM THE DATAPATH WAIT QUEUE.
                        0150   431 ;       IF A DRIVER PROCESS IS WAITING, THEN THE BUFFERED DATAPATH IS
                        0150   432 ;       ASSIGNED TO THE ASSOCIATED I/O CHANNEL AND THE DRIVER IS CALLED
                        0150   433 ;       VIA A JSB TO ITS BUFFERED DATAPATH WAIT RETURN ADDRESS.
                        0150   434 ;-
                        0150   435
                        0150   436 IOC$RELDATAP::                               ;RELEASE BUFFERED DATAPATH
 50   20 A5    D0      0150   437          MOVL    UCB$L_CRB(R5),R0              ;GET ADDRESS OF CRB
 51   28 A0    D0      0154   438          MOVL    CRB$L_INTD+VEC$L_ADP(R0),R1 ;GET ADDRESS OF ADP
 52   27 A0    98      0158   439          CVTBL   CRB$L_INTD+VEC$B_DATAPATH(R0),R2 ;GET DATAPATH DESIGNATOR
           29   19      015C   440          BLSS    10$                           ;IF LSS PERMANENT ASSIGNMENT
      00   00   F0      015E   441          INSV    #0,#VEC$V_DATAPATH,-          ;ZERO DATAPATH NUMBER
           05           0161   442                  #VEC$S_DATAPATH,-             ;
        27 A0           0162   443                  CRB$L_INTD+VEC$B_DATAPATH(R0)
           00   EF      0164   444          EXTZV   #VEC$V_DATAPATH,-             ;EXTRACT DATAPATH NUMBER
 52   52   05           0169   445                  #VEC$S_DATAPATH,R2,R2         ;
 50   14 B1    0F      0169   446          REMQUE  @ADP$L_DPQFL(R1),R0           ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
           19   1D      016D   447          BVS     20$                           ;IF VS NO DRIVER PROCESS WAITING
           38   BB      016F   448          PUSHR   #^M<R3,R4,R5>                 ;SAVE CONTEXT OF CURRENT DRIVER PROCESS
      55   50   D0      0171   449          MOVL    R0,R5                         ;COPY ADDRESS OF DRIVER PROCESS FORK BLOCK
 51   20 A5    D0      0174   450          MOVL    UCB$L_CRB(R5),R1              ;GET ADDRESS OF CRB
      00   52   F0      0178   451          INSV    R2,#VEC$V_DATAPATH,-          ;STORE ASSIGNED
           05           017B   452                  #VEC$S_DATAPATH,-             ; DATAPATH NUMBER
        27 A1           017C   453                  CRB$L_INTD+VEC$B_DATAPATH(R1)
 53   10 A5    7D      017E   454          MOVQ    UCB$L_FR3(R5),R3              ;RESTORE DRIVER PROCESS CONTEXT
      0C B5    16      0182   455          JSB     @UCB$L_FPC(R5)                ;CALL DRIVER AT DATAPATH WAIT RETURN ADDRESS
           38   BA      0185   456          POPR    #^M<R3,R4,R5>                 ;RESTORE PREVIOUS DRIVER PROCESS CONTEXT
                05      0187   457  10$:    RSB                                   ;
FA 24 A1   52   E3      0188   458  20$:    BBCS    R2,ADP$W_DPBITMAP(R1),10$ ;SET DATAPATH BIT AND EXIT
                        018D   459          BUG_CHECK INCONSTATE                  ;INCONSISTENT STATE
                05      019F   460          RSB                                   ;
```

```
                                01A0   462          .SBTTL  REQUEST BUFFERED DATAPATH
                                01A0   463 ;+
                                01A0   464 ; IOC$REQDATAP - REQUEST BUFFERED DATAPATH (WAIT)
                                01A0   465 ; IOC$REQDATAPNW - REQUEST BUFFERED DATAPATH (NO WAIT)
                                01A0   466 ;
                                01A0   467 ; THIS ROUTINE IS CALLED TO REQUEST A UNIBUS ADAPTER BUFFERED DATAPATH TO
                                01A0   468 ; PERFORM AN I/O OPERATION ON.
                                01A0   469 ;
                                01A0   470 ; INPUTS:
                                01A0   471 ;
                                01A0   472 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                01A0   473 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                01A0   474 ;
                                01A0   475 ;       IT IS ASSUMED THAT THE CALLER OWNS THE I/O CHANNEL ON WHICH THE
                                01A0   476 ;       TRANSFER IS TO OCCUR.
                                01A0   477 ;
                                01A0   478 ; OUTPUTS:
                                01A0   479 ;
                                01A0   480 ;       IF A BUFFERED DATAPATH HAS BEEN PERMANENTLY ASSIGNED TO THE ASSOCIATED
                                01A0   481 ;       I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                                01A0   482 ;       ELSE AN ATTEMPT IS MADE TO FIND A FREE BUFFERED DATAPATH. IF A FREE
                                01A0   483 ;       BUFFERED DATAPATH IS FOUND, THEN ITS NUMBER IS STORED IN THE ASSO-
                                01A0   484 ;       CIATED CHANNEL REQUEST BLOCK AND CONTROL IS RETURNED TO THE CALLER.
                                01A0   485 ;       ELSE IF NO WAIT IS SPECIFIED, THEN A FAILURE INDICATION IS RETURNED
                                01A0   486 ;       TO THE CALLER. ELSE THE DRIVER PROCESS CONTEXT IS SAVED IN ITS FORK
                                01A0   487 ;       BLOCK, THE FORK BLOCK IS INSERTED IN THE BUFFERED DATAPATH WAIT QUEUE,
                                01A0   488 ;       AND A RETURN TO THE DRIVER PROCESS' CALLER IS EXECUTED.
                                01A0   489 ;-
                                01A0   490
                                01A0   491          .ENABL  LSB
                                01A0   492 IOC$REQDATAPNW::                             ;REQUEST BUFFERED DATAPATH NO WAIT
                00    DD        01A0   493          PUSHL   #0                          ;SET NO WAIT INDICATOR
                02    11        01A2   494          BRB     5$                          ;
                                01A4   495 IOC$REQDATAP::                               ;REQUEST BUFFERED DATAPATH WAIT
                01    DD        01A4   496          PUSHL   #1                          ;SET WAIT INDICATOR
          50    20    A5  DO    01A6   497 5$:      MOVL    UCB$L_CRB(R5),R0            ;GET ADDRESS OF CRB
28 27 A0        07    E0        01AA   498          BBS     #VEC$V_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R0),20$ ;IF SET, PERMANENT
          51    28    A0  DO    01AF   499          MOVL    CRB$L_INTD+VEC$L_ADP(R0),R1 ;GET ADDRESS OF ADP
                10    00  EA    01B3   500          FFS     #0,#ADP$C_NUMDATAP,ADP$W_DPBITMAP(R1),R2 ;SEARCH FOR FREE DATAPATH
          52    24    A1        01B6
                08    13        01B9   501          BEQL    10$                         ;IF EQL NONE FOUND
                00    52  F0    01BB   502          INSV    R2,#VEC$V_DATAPATH,-        ;STORE ASSIGNED
                05              01BE   503                  #VEC$S_DATAPATH,-       ; DATAPATH NUMBER
                27    A0        01BF   504                  CRB$L_INTD+VEC$B_DATAPATH(R0)
11 24 A1        52    E4        01C1   505          BBSC    R2,ADP$W_DPBITMAP(R1),20$ ;CLEAR DATAPATH BIT AND EXIT
                13    8E  E9    01C6   506 10$:     BLBC    (SP)+,30$                   ;IF LBC NO WAIT SPECIFIED
          10 A5 53    7D        01C9   507          MOVQ    R3,UCB$L_FR3(R5)            ;SAVE DRIVER PROCESS CONTEXT
                0C    A5  8EDO  01CD   508          POPL    UCB$L_FPC(R5)               ;SAVE DATAPATH WAIT RETURN ADDRESS
          18 B1 65    0E        01D1   509          INSQUE  UCB$L_FQFL(R5),@ADP$L_DPQBL(R1) ;INSERT DRIVER PROCESS IN DATAPATH W
                7E    D5        01D5   510          TSTL    -(SP)                       ;PUSH DUMMY INDICATOR ON STACK
                50    00'   9A  01D7   511 20$:     MOVZBL  S^#SS$_NORMAL,R0            ;SET SUCCESS INDICATOR
                8E    D5        01DA   512          TSTL    (SP)+                       ;REMOVE WAIT INDICATOR FROM STACK
                      05        01DC   513 30$:     RSB                                 ;
                                01DD   514          .DSABL  LSB
```

```
                        01DD   516          .SBTTL  RELEASE UNIBUS MAP REGISTERS
                        01DD   517 ;+
                        01DD   518 ; IOC$RELMAPREG - RELEASE UNIBUS MAP REGISTERS
                        01DD   519 ;
                        01DD   520 ; THIS ROUTINE IS CALLED TO RELEASE UNIBUS MAP REGISTERS THAT WERE PREVIOUSLY
                        01DD   521 ; ASSIGNED FOR AN I/O TRANSFER.
                        01DD   522 ;
                        01DD   523 ; INPUTS:
                        01DD   524 ;
                        01DD   525 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                        01DD   526 ;
                        01DD   527 ;       IT IS ASSUMED THAT THE CALLER STILL OWNS THE I/O CHANNEL ON WHICH
                        01DD   528 ;       THE TRANSFER TOOK PLACE.
                        01DD   529 ;
                        01DD   530 ; OUTPUTS:
                        01DD   531 ;
                        01DD   532 ;       IF THE MAPPING REGISTERS HAVE BEEN PERMANENTLY ASSIGNED TO THE ASSO-
                        01DD   533 ;       CIATED I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                        01DD   534 ;       ELSE THE MAPPING REGISTERS ARE RELEASED AND AN ATTEMPT IS MADE TO
                        01DD   535 ;       REMOVE THE NEXT DRIVER PROCESS FROM THE MAP REGISTER WAIT QUEUE. IF
                        01DD   536 ;       A DRIVER PROCESS IS WAITING, THEN IT IS REMOVED FROM THE MAP REGISTER
                        01DD   537 ;       WAIT QUEUE AND AN ATTEMPT IS MADE TO ASSIGN THE REQUESTED NUMBER OF
                        01DD   538 ;       CONTIGUOUS MAP REGISTERS TO THE TRANSFER. IF THE ALLOCATION IS SUCCESS-
                        01DD   539 ;       FUL, THEN THE DRIVER IS CALLED VIA A JSB TO ITS MAP REGISTER WAIT
                        01DD   540 ;       RETURN ADDRESS. WHEN THE DRIVER RETURNS ANOTHER ATTEMPT WILL BE MADE
                        01DD   541 ;       TO ALLOCATE MAP REGISTERS TO THE NEXT DRIVER PROCESS IN THE MAP REGISTER
                        01DD   542 ;       WAIT QUEUE. THE PROCESS OF ATTEMPTING TO ALLOCATE REGISTERS TO WAIT-
                        01DD   543 ;       ING DRIVER PROCESSES IS CONTINUED UNTIL EITHER THERE ARE NO DRIVER
                        01DD   544 ;       PROCESSES WAITING OR AN ALLOCATION FAILURE OCCURS. IN THE CASE OF
                        01DD   545 ;       AN ALLOCATION FAILURE THE DRIVER PROCESS IS REINSERTED AT THE FRONT
                        01DD   546 ;       OF THE MAP REGISTER WAIT QUEUE.
                        01DD   547 ;-
                        01DD   548
                        01DD   549 IOC$RELMAPREG::                              ;RELEASE UNIBUS MAP REGISTERS
   51    20 A5    D0    01DD   550          MOVL    UCB$L_CRB(R5),R1            ;GET ADDRESS OF CRB
30 24 A1    0F    E0    01E1   551          BBS     #VEC$V_MAPLOCK,CRB$L_INTD+VEC$W_MAPREG(R1),40$ ;IF SET, PERMANENT
     0078 8F    BB    01E6   552          PUSHR   #^M<R3,R4,R5,R6>            ;SAVE REGISTERS
   52    28 A1    D0    01EA   553          MOVL    CRB$L_INTD+VEC$L_ADP(R1),R2 ;GET ADDRESS OF ADP
      56    52    D0    01EE   554          MOVL    R2,R6                       ;SAVE ADDRESS OF ADP
   54    24 A1    3C    01F1   555          MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;GET STARTING MAP REGISTER NUMBER
      50    00    D2    01F5   556          MCOML   #0,R0                       ;SET ALTER PATTERN
            2F    10    01F8   557          BSBB    IOC$ALTUBAMAP               ;ALTER MAP REGISTER BITMAP
   55    1C B6    0F    01FA   558 10$:      REMQUE  @ADP$L_MRQFL(R6),R5         ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
            12    1D    01FE   559          BVS     30$                         ;IF VS NO DRIVER PROCESS WAITING
            49    10    0200   560          BSBB    IOC$ALOUBAMAP               ;SEARCH MAP REGISTER BITMAP AND ALLOCATE
         09 50    E9    0202   561          BLBC    R0,20$                      ;IF LBC ALLOCATION FAILURE
   53    10 A5    7D    0205   562          MOVQ    UCB$L_FR3(R5),R3            ;RESTORE DRIVE PROCESS CONTEXT
         0C B5    16    0209   563          JSB     @UCB$L_FPC(R5)              ;CALL DRIVER AT MAP REGISTER WAIT RETURN ADD
            EC    11    020C   564          BRB     10$                         ;
   1C A6    65    0E    020E   565 20$:      INSQUE  UCB$L_FQFL(R5),ADP$L_MRQFL(R6) ;REINSERT DRIVER PROCESS AT FRONT OF
      0078 8F    BA    0212   566 30$:      POPR    #^M<R3,R4,R5,R6>            ;RESTORE REGISTERS
                  05    0216   567 40$:      RSB                                 ;
```

```
                                0217   569          .SBTTL  REQUEST UNIBUS MAP REGISTERS
                                0217   570 ;+
                                0217   571 ; IOC$REQMAPREG - REQUEST UNIBUS MAP REGISTERS
                                0217   572 ;
                                0217   573 ; THIS ROUTINE IS CALLED TO REQUEST UNIBUS MAP REGISTERS TO PERFORM AN
                                0217   574 ; I/O TRANSFER.
                                0217   575 ;
                                0217   576 ; INPUTS:
                                0217   577 ;
                                0217   578 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0217   579 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                0217   580 ;
                                0217   581 ;       IT IS ASSUMED THAT THE CALLER OWNS THE I/O CHANNEL ON WHICH THE
                                0217   582 ;       TRANSFER IS TO OCCUR ON.
                                0217   583 ;
                                0217   584 ; OUTPUTS:
                                0217   585 ;
                                0217   586 ;       IF MAP REGISTERS HAVE BEEN PERMANENTLY ASSIGNED TO THE ASSOCIATED
                                0217   587 ;       I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                                0217   588 ;       ELSE AN ATTEMPT IS MADE TO ALLOCATE THE REQUESTED NUMBER OF MAP REG-
                                0217   589 ;       ISTERS. IF SUFFICIENT CONTIGUOUS MAP REGISTERS ARE FOUND, THEN THEY
                                0217   590 ;       ARE ASSIGNED TO THE ASSOCIATED I/O CHANNEL AND CONTROL IS RETURNED
                                0217   591 ;       TO THE CALLER. ELSE THE DRIVER PROCESS CONTEXT IS SAVED IN ITS FORK
                                0217   592 ;       BLOCK, THE FORK BLOCK IS INSERTED IN THE MAP REGISTER WAIT QUEUE,
                                0217   593 ;       AND A RETURN TO THE DRIVER PROCESS' CALLER IS EXECUTED.
                                0217   594 ;-
                                0217   595
                                0217   596 IOC$REQMAPREG::                             ;REQUEST UNIBUS MAP REGISTERS
      32   10                   0217   597          BSBB    IOC$ALOUBAMAP              ; ALLOCATE UBA MAP REGISTER
   0C 50   E8                   0219   598          BLBS    R0,10$                     ;IF LBS SUCCESSFUL ALLOCATION
10 A5 53   7D                   021C   599          MOVQ    R3,UCB$L_FR3(R5)           ;SAVE DRIVER PROCESS CONTEXT
   0C A5 8ED0                   0220   600          POPL    UCB$L_FPC(R5)              ;SAVE MAP REGISTER WAIT RETURN ADDRESS
20 B2 65   0E                   0224   601          INSQUE  UCB$L_FQFL(R5),@ADP$L_MRQBL(R2) ;INSERT PROCESS IN MAP REGISTER WAIT
      05                        0228   602 10$:     RSB                                ;
```

```
                                0229   604          .SBTTL  ALTER UBA MAP REGISTER BITMAP
                                0229   605 ;+
                                0229   606 ; IOC$ALTUBAMAP - ALTER UBA MAP REGISTER BIT MAP
                                0229   607 ;
                                0229   608 ; THIS ROUTINE IS CALLED TO EITHER CLEAR OF SET A FILLD OF BITS IN THE UBA MAP
                                0229   609 ; REGISTER ALLOCATION BITMAP.
                                0229   610 ;
                                0229   611 ; INPUTS:
                                0229   612 ;
                                0229   613 ;       R0 = ALTERATION BIT MASK.
                                0229   614 ;       R1 = ADDRESS OF CRB.
                                0229   615 ;       R2 = ADDRESS OF ADP.
                                0229   616 ;       R4 = STARTING MAP REGISTER NUMBER.
                                0229   617 ;
                                0229   618 ; OUTPUTS:
                                0229   619 ;
                                0229   620 ;       THE SPECIFIED BIT FIELD IN THE UBA MAP ALLOCATION BIT MAP IS EITHER SET
                                0229   621 ;       OR CLEARED DEPENDING ON THE STATE OF THE ALTERATION MASK.
                                0229   622 ;
                                0229   623 ;       R3 AND R4 ARE DESTROYED.
                                0229   624 ;-
                                0229   625
                                0229   626 IOC$ALTUBAMAP::                          ;ALTER MAP REGISTER BIT MAP
          53    26 A1    9A     0229   627          MOVZBL  CRB$L_INTD+VEC$B_NUMREG(R1),R3 ;GET NUMBER OF BITS TO ALTER
                53    20    D1   022D   628 10$:     CMPL    #32,R3                  ;MORE THAN LONGWORD LEFT?
                      0E    18   0230   629          BGEQ    20$                     ;IF GEQ NO
          20    54    50    F0   0232   630          INSV    R0,R4,#32,ADP$W_MRBITMAP(R2) ;ALTER BITMAP WITH SUPPLIED PATTERN
                   26 A2         0236
                54    20    C0   0238   631          ADDL    #32,R4                  ;UPDATE STARTING BIT POSITION
                53    20    C2   023B   632          SUBL    #32,R3                  ;REDUCE NUMBER OF BITS TO ALTER
                      ED    11   023E   633          BRB     10$                     ;
          53    54    50    F0   0240   634 20$:     INSV    R0,R4,R3,ADP$W_MRBITMAP(R2) ;ALTER BITMAP WITH SUPPLIED PATTERN
                   26 A2         0244
                      05    0246   635          RSB                             ;
```

```
                                0247   637          .SBTTL  SEARCH MAP REGISTER BITMAP AND ALLOCATE MAP REGISTERS
                                0247   638 ;+
                                0247   639 ; IOC$ALOUBAMAP - ALLOCATE UBA MAP REGISTERS (CRB DATABASE SPECIFIED)
                                0247   640 ; IOC$ALOUBAMAPN - ALLOCATE UBA MAP REGISTERS (ARGUMENT SPECIFIED)
                                0247   641 ;
                                0247   642 ; THIS ROUTINE IS CALLED TO ALLOCATE UBA MAP REGISTERS AND TO MARK THE ALLOCATION
                                0247   643 ; IN THE UBA MAP REGISTER ALLOCATION BITMAP.
                                0247   644 ;
                                0247   645 ; INPUTS:
                                0247   646 ;
                                0247   647 ;       R3 = NUMBER OF MAP REGISTERS TO ALLOCATE (ARGUMENT SPECIFIED ENTRY).
                                0247   648 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0247   649 ;
                                0247   650 ; OUTPUTS:
                                C247   651 ;
                                0247   652 ;       R0 = SUCCESS INDICATION.
                                0247   653 ;-
                                0247   654
                                0247   655          .ENABL  LSB
                                0247   656 IOC$ALOUBAMAPN::                           ;ALLOCATE UBA MAP REGISTERS ARGUMENT SPECIFI
                     38   BB    0247   657          PUSHR   #^M<R3,R4,R5>            ;SAVE REGISTERS
                     15   11    0249   658          BRB     5$                       ;
                                024B   659 IOC$ALOUBAMAP::                           ;ALLOCATE UBA MAP REGISTERS CRB SPECIFIED
                     38   BB    024B   660          PUSHR   #^M<R3,R4,R5>            ;SAVE REGISTERS
                 53   6E A5    3C    024D   661          MOVZWL  UCB$W_BCNT(R5),R3        ;GET TRANSFER BYTE COUNT
                 54   6C A5    3C    0251   662          MOVZWL  UCB$W_BOFF(R5),R4        ;GET BYTE OFFSET IN PAGE
             53    03FF C344    9E    0255   663          MOVAB   ^X3FF(R3)[R4],R3         ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
                 53   F7 8F    78    025B   664          ASHL    #-9,R3,R3                ;CALCULATE NUMBER OF MAP REGISTERS REQUIRED
                      53          025F
                      50   D4    0260   665 5$:       CLRL    R0                       ;ASSUME ALLOCATION FAILURE
                51    20 A5    D0    0262   666          MOVL    UCB$L_CRB(R5),R1         ;GET ADDRESS OF CRB
                52    28 A1    D0    0266   667          MOVL    CRB$L_INTD+VEC$L_ADP(R1),R2 ;GET ADDRESS OF ADP
             39 24 A1    0F    E0    026A   668          BBS     #VEC$V_MAPLOCK,CRB$L_INTD+VEC$W_MAPREG(R1),40$ ;IF SET, PERMANENT
                26 A1    53    90    026F   669          MOVB    R3,CRB$L_INTD+VEC$B_NUMREG(R1) ;SET NUMBER OF MAP REGISTERS ALLOCATE
                      54   D4    0273   670          CLRL    R4                       ;CLEAR STARTING BIT POSITION
              55    54    53    C1    0275   671 10$:      ADDL3   R3,R4,R5                 ;CALCULATE HIGHEST BIT IN REQUIRED SCAN
             01F0 8F    55    B1    0279   672          CMPW    R5,#496                  ;BEYOND END OF ALLOCATION BITMAP?
                      2A   14    027E   673          BGTR    50$                      ;IF GTR YES
                 20   54    EA    0280   674          FFS     R4,#32,ADP$W_MRBITMAP(R2),R4 ;FIND A SET BIT
                54    26 A2          0283
                      ED   13    0286   675          BEQL    10$                      ;IF EQL BIT NOT FOUND
              55    54    53    C1    0288   676          ADDL3   R3,R4,R5                 ;CALCULATE HIGH BIT FOR SUCCESSFUL ALLOCATIO
                24 A1    54    B0    028C   677          MOVW    R4,CRB$L_INTD+VEC$W_MAPREG(R1) ;SAVE STARTING BIT NUMBER
                 20   54    EB    0290   678 20$:      FFC     R4,#32,ADP$W_MRBITMAP(R2),R4 ;FIND A CLEAR BIT
                54    26 A2          0293
                 55   54    D1    0296   679          CMPL    R4,R5                    ;ENOUGH SET BITS SCANNED OVER?
                      07   18    0299   680          BGEQ    30$                      ;IF GEQ YES
             F0 26 A2    54    E0    029B   681          BBS     R4,ADP$W_MRBITMAP(R2),20$ ;IF SET, CONTINUE SCAN
                      D3   11    02A0   682          BRB     10$                      ;
                54    24 A1    3C    02A2   683 30$:      MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;RETRIEVE STARTING MAP REGISTER
                      81   10    02A6   684          BSBB    IOC$ALTUBAMAP            ;ALTER MAP REGISTER BITMAP
                      50   D6    02A8   685 40$:      INCL    R0                       ;SET SUCCESS INDICATOR
                      38   BA    02AA   686 50$:      POPR    #^M<R3,R4,R5>            ;RESTORE REGISTERS
                      05   02AC   687          RSB                              ;
                           02AD   688          .DSABL  LSB
```

```
                     02AD   690          .SBTTL  RETURN TO CALLER
                     02AD   691 ;+
                     02AD   692 ; IOC$RETURN - RETURN TO CALLER
                     02AD   693 ;
                     02AD   694 ; THIS ROUTINE IS CALLED AS A RESULT OF A DDT DISPATCH TO A NULL ENTRY. ITS
                     02AD   695 ; FUNCTION IS MERELY TO RETURN TO ITS CALLER.
                     02AD   696 ;
                     02AD   697 ; INPUTS:
                     02AD   698 ;
                     02AD   699 ;       NONE.
                     02AD   700 ;
                     02AD   701 ; OUTPUTS:
                     02AD   702 ;
                     02AD   703 ;       NONE.
                     02AD   704 ;-
                     02AD   705
                     02AD   706 IOC$RETURN::                         ;RETURN TO CALLER
                  05 02AD   707          RSB                         ;
```

```
                                02AE    709          .SBTTL  WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
                                02AE    710 ;+
                                02AE    711 ; IOC$WFIKPCH - WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
                                02AE    712 ;
                                02AE    713 ; THIS ROUTINE IS CALLED TO SOFTWARE ENABLE INTERRUPTS AND TIMEOUT ON
                                02AE    714 ; A DEVICE UNIT AND TO KEEP THE CHANNEL. THIS ROUTINE CAN BE CALLED AT
                                02AE    715 ; EITHER FORK OR DEVICE INTERRUPT LEVEL.
                                02AE    716 ;
                                02AE    717 ; INPUTS:
                                02AE    718 ;
                                02AE    719 ;       00(SP) = RETURN ADDRESS OF CALLER.
                                02AE    720 ;       04(SP) = TIMEOUT VALUE IN SECONDS.
                                02AE    721 ;       08(SP) = IPL TO LOWER TO AFTER SETTING WAIT.
                                02AE    722 ;       12(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                02AE    723 ;
                                02AE    724 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                02AE    725 ;
                                02AE    726 ; OUTPUTS:
                                02AE    727 ;
                                02AE    728 ;       THE TIMEOUT VALUE IS COMPUTED AND STORED IN DUE TIME, REGISTERS R3 AND
                                02AE    729 ;       R4 ALONG WITH THE RETURN PC ARE SAVED IN THE FORK BLOCK, INTERRUPTS AND
                                02AE    730 ;       TIMEOUT ARE ENABLED, AND A RETURN TO THE CALLER'S CALLER IS EXECUTED.
                                02AE    731 ;-
                                02AE    732
                                02AE    733 IOC$WFIKPCH::                               ;WAITFOR INTERRUPT/TIMEOUT AND KEEP CHANNEL
                   6E   02   C0 02AE    734          ADDL    #2,(SP)                    ;CALCULATE OFFSET TO NORMAL RETURN
                10 A5   53   7D 02B1    735          MOVQ    R3,UCB$L_FR3(R5)           ;SAVE REGISTERS R3 AND R4
                    0C A5 8ED0 02B5    736          POPL    UCB$L_FPC(R5)              ;SAVE INTERRUPT RETURN ADDRESS
                58 A5   03   A8 02B9    737          BISW    #UCB$M_INT!UCB$M_TIM,UCB$W_STS(R5) ;ENABLE INTERRUPT AND TIMEOUT
02                      8E   C1 02BD     .1          ADDL3   (SP)+,EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ;SET TIMEOUT TIME
              00000000'EF        02BF
                   5C A5         02C4
-1                0040 8F   AA  02C6    739          BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR UNIT TIMED OUT
                   58 A5         02CA
                                 02CC    740          ENBINT                          ;ENABLE INTERRUPTS
                         05      02CF    741          RSB                             ;
```

```
                                02D0    743          .SBTTL  WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
                                02D0    744 ;+
                                02D0    745 ; IOC$WFIRLCH - WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
                                02D0    746 ;
                                02D0    747 ; THIS ROUTINE IS CALLED TO SOFTWARE ENABLE INTERRUPTS AND TIMEOUT ON A DEVICE
                                02D0    748 ; UNIT AND TO RELEASE THE CHANNEL. THIS ROUTINE CAN ONLY BE CALLED AT FORK LEVEL.
                                02D0    749 ;
                                02D0    750 ; INPUTS:
                                02D0    751 ;
                                02D0    752 ;       00(SP) = RETURN ADDRESS OF CALLER.
                                02D0    753 ;       04(SP) = TIMEOUT VALUE IN SECONDS.
                                02D0    754 ;       08(SP) = IPL TO LOWER TO AFTER SETTING WAIT.
                                02D0    755 ;       12(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                02D0    756 ;
                                02D0    757 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                02D0    758 ;
                                02D0    759 ; OUTPUTS:
                                02D0    760 ;
                                02D0    761 ;       THE TIMEOUT VALUE IS COMPUTED AND STORED IN DUE TIME, REGISTERS R3 AND
                                02D0    762 ;       R4 ALONG WITH THE RETURN PC ARE SAVED IN THE FORK BLOCK, INTERRUPTS AND
                                02D0    763 ;       TIMEOUT ARE ENABLED, THE CHANNEL IS RELEASED, AND A RETURN TO THE CALLER'S
                                02D0    764 ;       CALLER IS EXECUTED.
                                02D0    765 ;-
                                02D0    766
                                02D0    767 IOC$WFIRLCH::                               ;WAITFOR INTERRUPT/TIMEOUT AND RELEASE CHANN
          6E    02    C0        02D0    768          ADDL    #2,(SP)                    ;CALCULATE OFFSET TO NORMAL RETURN
        10 A5    53    7D        02D3    769          MOVQ    R3,UCB$L_FR3(R5)           ;SAVE REGISTERS R3 AND R4
           0C A5  8ED0           02D7    770          POPL    UCB$L_FPC(R5)              ;SAVE INTERRUPT RETURN ADDRESS
        58 A5    03    A8        02DB    771          BISW    #UCB$M_INT!UCB$M_TIM,UCB$W_STS(R5) ;ENABLE INTERRUPT AND TIMEOUT
02                  8E    C1    02DF     .1          ADDL3   (SP)+,EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ;SET TIMEOUT TIME
                00000000'EF       02E1
                   5C A5          02E6
-1                0040 8F    AA   02E8    773          BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR UNIT TIMED OUT
                   58 A5          02EC
                                  02EE    774          ENBINT                          ;ENABLE INTERRUPTS
                      FD5E    31  02F1    775          BRW     IOC$RELCHAN             ;RELEASE ALL CHANNELS AND RETURN TO CALLER
                                  02F4    776
                                  02F4    777
```

```
                                02F4    779          .SBTTL  ALLOCATE SYSTEM PAGE TABLE
                                02F4    780 ;+
                                02F4    781 ; IOC$ALLOSPT - ALLOCATE SYSTEM PAGE TABLE
                                02F4    782 ;
                                02F4    783 ; THIS ROUTINE ALLOCATES SYSTEM PAGE TABLE (SPT) ENTRIES.
                                02F4    784 ;
                                02F4    785 ; INPUTS:
                                02F4    786 ;
                                02F4    787 ;       R1 = NUMBER OF SPT ENTRIES TO BE ALLOCATED
                                02F4    788 ;
                                02F4    789 ;       BOO$GL_SPTFREL = LOWEST FREE VPN
                                02F4    790 ;       BOO$GL_SPTFREH = HIGHEST FREE VPN
                                02F4    791 ;
                                02F4    792 ;       IT IS ASSUMED THAT THE CALLER IS RUNNING AT IPL$_SYNCH.
                                02F4    793 ;
                                02F4    794 ; OUTPUTS:
                                02F4    795 ;
                                02F4    796 ;       R0 = SUCCESS INDICATION.
                                02F4    797 ;       R2 = STARTING PAGE NUMBER ALLOCATED (SVPN).
                                02F4    798 ;       R3 = ADDRESS OF BASE OF SYSTEM PAGE TABLE (MMG$GL_SPTBASE).
                                02F4    799 ;
                                02F4    800 ;       R1 IS PRESERVED ACROSS CALL.
                                02F4    801 ;-
                                02F4    802 IOC$ALLOSPT::                               ;ALLOCATE SYSTEM PAGE TABLE
03          00000000'EF   D0    02F4     .1         MOVL    L^MMG$GL_SPTBASE,R3        ; Get addr of base of SPT
                    53          02FA
03               50   01   D0   02FB     .2         MOVL    #1,R0                      ; Set success
-8                             02FE    811 10$:
                          05   02FE    812         RSB                                ;
                               02FF    813
                               02FF    814         .END
```

```
ADP$B_NUMBER      0000000B    D       DDT$L_REGDUMP     00000010    D       IRP$L_IOQBL       00000004    D
ADP$B_PORT        00000020    D       DDT$L_START       00000000    D       IRP$L_IOQFL       00000000    D
ADP$B_TYPE        0000000A    D       DDT$L_UNITINIT    00000018    D       IRP$L_IOSB        00000024    D
ADP$C_DRADPLEN    00000014    D       DDT$L_UNSOLINT    00000004    D       IRP$L_IOST1       00000034    D
ADP$C_MBAADPLEN   00000014    D       DDT$W_DIAGBUF     00000014    D       IRP$L_IOST2       00000038    D
ADP$C_MPMADPLEN   00000070    D       DDT$W_ERRORBUF    00000016    D       IRP$L_MEDIA       00000034    D
ADP$C_NUMDATAP =  00000010    D       EMB$B_DV_ERTCNT=  00000010    D       IRP$L_PID         0000000C    D
ADP$C_UBAADPLEN   00000070    D       EMB$Q_DV_IOSB   = 00000012    D       IRP$L_SEGVBN      00000040    D
ADP$K_DRADPLEN    00000014    D       EMB$W_DV_STS    = 0000001A    D       IRP$L_SEQNUM      00000048    D
ADP$K_MBAADPLEN   00000014    D       ERL$RELEASEMB     ********    X   02  IRP$L_SVAPTE      0000002C    D
ADP$K_MPMADPLEN   00000070    D       EXE$GL_ABSTIM     ********    X   02  IRP$L_TT_TERM     00000038    D
ADP$K_UBAADPLEN   00000070    D       EXE$GQ_SYSTIME    ********    X   02  IRP$L_UCB         0000001C    D
ADP$L_CRB         00000010    D       IDB$B_TYPE        0000000A    D       IRP$L_WIND        00000018    D
ADP$L_CSR         00000000    D       IDB$C_LENGTH      00000034    D       IRP$Q_NT_PRVMSK   0000003C    D
ADP$L_DPQBL       00000018    D       IDB$K_LENGTH      00000034    D       IRP$V_DIAGBUF   = 00000007    D
ADP$L_DPQFL       00000014    D       IDB$L_ADP         00000010    D       IRP$W_ABCNT       0000003C    D
ADP$L_INTD        00000064    D       IDB$L_CSR         00000000    D       IRP$W_BCNT        00000032    D
ADP$L_LINK        00000004    D       IDB$L_OWNER       00000004    D       IRP$W_BOFF        00000030    D
ADP$L_MRQBL       00000020    D       IDB$L_UCBLST      00000014    D       IRP$W_CHAN        00000028    D
ADP$L_MRQFL       0000001C    D       IDB$W_SIZE        00000008    D       IRP$W_FUNC        00000020    D
ADP$L_PRQQBL      00000018    D       IDB$W_UNITS       0000000C    D       IRP$W_OBCNT       0000003E    D
ADP$L_PRQQFL      00000014    D       IOC$ALLOSPT       000002F4 RG D   02  IRP$W_SIZE        00000008    D
ADP$L_SHB         0000001C    D       IOC$ALOUBAMAP     00000248 RG D   02  IRP$W_STS         0000002A    D
ADP$L_VECTOR      00000010    D       IOC$ALOUBAMAPN    00000247 RG D   02  IRP$W_TT_PRMPT    0000003C    D
ADP$W_ADPTYPE     0000000E    D       IOC$ALTUBAMAP     00000229 RG D   02  MMG$GL_SPTBASE    ********    X   02
ADP$W_DPBITMAP    00000024    D       IOC$CANCELIO      00000000 RG D   02  PCB$B_ASTACT      0000000C    D
ADP$W_MRBITMAP    00000026    D       IOC$DIAGBUFILL    00000017 RG D   02  PCB$B_ASTEN       0000000D    D
ADP$W_SIZE        00000008    D       IOC$GL_PSBL       ********    X   02  PCB$B_PRI         0000000B    D
ADP$W_TR          0000000C    D       IOC$INITIATE      0000011D RG D   02  PCB$B_PRIB        0000002F    D
BUG$CHECK         ********    X   02  IOC$RELCHAN       00000052 RG D   02  PCB$B_TYPE        0000000A    D
CRB$B_MASK        0000000E    D       IOC$RELDATAP      00000150 RG D   02  PCB$B_WEFC        0000002E    D
CRB$B_TYPE        0000000A    D       IOC$RELMAPREG     000001DD RG D   02  PCB$C_LENGTH      0000008C    D
CRB$C_LENGTH      00000038    D       IOC$RELSCHAN      00000048 RG D   02  PCB$K_LENGTH      0000008C    D
CRB$K_LENGTH      00000038    D       IOC$REQCOM        000000DD RG D   02  PCB$L_ARB         00000084    D
CRB$L_INTD        00000014    D       IOC$REQDATAP      000001A4 RG D   02  PCB$L_ASTQBL      00000014    D
CRB$L_INTD2       00000038    D       IOC$REQDATAPNW    000001A0 RG D   02  PCB$L_ASTQFL      00000010    D
CRB$L_LINK        00000010    D       IOC$REQMAPREG     00000217 RG D   02  PCB$L_EFC2P       00000058    D
CRB$L_WQBL        00000004    D       IOC$REQPCHANH     000000A9 RG D   02  PCB$L_EFC3P       0000005C    D
CRB$L_WQFL        00000000    D       IOC$REQPCHANL     000000B2 RG D   02  PCB$L_EFCS        00000050    D
CRB$M_BSY      == 00000001    D       IOC$REQSCHANH     00000095 RG D   02  PCB$L_EFCU        00000054    D
CRB$V_BSY      == 00000000    D       IOC$REQSCHANL     0000009F RG D   02  PCB$L_EFWM        0000004C    D
CRB$W_REFC        0000000C    D       IOC$RETURN        000002AD RG D   02  PCB$L_JIB         00000078    D
CRB$W_SIZE        00000008    D       IOC$WFIKPCH       000002AE RG D   02  PCB$L_OWNER       0000001C    D
DDB$B_ACPCLASS    00000013    D       IOC$WFIRLCH       000002D0 RG D   02  PCB$L_PHD         00000064    D
DDB$B_TYPE        0000000A    D       IPL$_IOPOST    =  00000004    D       PCB$L_PHYPCB      00000018    D
DDB$C_LENGTH      00000034    D       IRP$B_CARCON      00000038    D       PCB$L_PID         00000060    D
DDB$K_LENGTH      00000034    D       IRP$B_EFN         00000022    D       PCB$L_PQB         0000004C    D
DDB$L_ACPD        00000010    D       IRP$B_PRI         00000023    D       PCB$L_SQBL        00000004    D
DDB$L_DDT         0000000C    D       IRP$B_RMOD        0000000B    D       PCB$L_SQFL        00000000    D
DDB$L_LINK        00000000    D       IRP$B_TYPE        0000000A    D       PCB$L_STS         00000024    D
DDB$L_UCB         00000004    D       IRP$C_LENGTH      0000005C    D       PCB$L_UIC         00000088    D
DDB$T_DRVNAME     00000024    D       IRP$K_LENGTH      0000005C    D       PCB$L_WSSWP       00000020    D
DDB$T_NAME        00000014    D       IRP$L_ARB         00000050    D       PCB$L_WTIME       00000028    D
DDB$W_SIZE        00000008    D       IRP$L_AST         00000010    D       PCB$Q_PRIV        0000007C    D
DDT$L_ALTSTART    0000001C    D       IRP$L_ASTPRM      00000014    D       PCB$T_LNAME       00000068    D
DDT$L_CANCEL      0000000C    D       IRP$L_DIAGBUF     00000044    D       PCB$T_TERMINAL    00000044    D
DDT$L_FDT         00000008    D       IRP$L_EXTEND      0000004C    D       PCB$W_APTCNT      00000030    D
```

```
PCB$W_ASTCNT      00000038     D
PCB$W_BIOCNT      0000003A     D
PCB$W_BIOLM       0000003C     D
PCB$W_DIOCNT      00C0003E     D
PCB$W_DIOLM       00000040     D
PCB$W_GPGCNT      00000034     D
PCB$W_GRP         0000008A     D
PCB$W_MEM         00000088     D
PCB$W_MTXCNT      0000000E     D
PCB$W_PPGCNT      00000036     D
PCB$W_PRCCNT      00000042     D
PCB$W_SIZE        00000008     D
PCB$W_STATE       0000002C     D
PCB$W_TMBU        00000032     D
PR$_IPL         = 00000012     D
PR$_SIRR        = 00000014     D
RELEASE           0000011A R   D   02
SIZ...          = 00000001     D
SS$_NORMAL        ********   X   02
UCB$B_AMOD        00000053     D
UCB$B_CEX         00000077     D
UCB$B_CM1         0000004A     D
UCB$B_CM2         0000004B     D
UCB$B_DEVCLASS    00000038     D
UCB$B_DEVTYPE     00000039     D
UCB$B_DIPL        00000052     D
UCB$B_DX_SCTCNT   000000A6     D
UCB$B_ERTCNT      00000070     D
UCB$B_ERTMAX      00000071     D
UCB$B_FEX         00000076     D
UCB$B_FIPL        0000000B     D
UCB$B_LOCSRV      0000003C     D
UCB$B_OFFNDX      00000094     D
UCB$B_OFFRTC      00000095     D
UCB$B_REMSRV      0000003D     D
UCB$B_SECTORS     0000003C     D
UCB$B_SLAVE       00000074     D
UCB$B_SPR         00000075     D
UCB$B_STATE       00000052     D
UCB$B_TRACKS      0000003D     D
UCB$B_TT_CRFILL   0000009D     D
UCB$B_TT_DECRF    000000A1     D
UCB$B_TT_DELFF    000000A2     D
UCB$B_TT_DESPEE   000000A0     D
UCB$B_TT_DETYPE   000000A4     D
UCB$B_TT_LFFILL   0000009E     D
UCB$B_TT_SPEED    0000009C     D
UCB$B_TYPE        0000000A     D
UCB$B_VERTSZ      0000003F     D
UCB$C_LENGTH      00000074     D
UCB$C_MB_LENGTH   00000090     D
UCB$C_TT_LENGTH   000000BC     D
UCB$K_LENGTH      00000074     D
UCB$K_MB_LENGTH   00000090     D
UCB$K_TT_LENGTH   000000BC     D
UCB$L_AMB         00000054     D
UCB$L_ASTQBL      00000010     D
UCB$L_ASTQFL      0000000C     D
UCB$L_CPID        0000005C     D
UCB$L_CRB         00000020     D
UCB$L_DDB         00000024     D
UCB$L_DEVCHAR     00000034     D
UCB$L_DEVDEPEND   0000003C     D
UCB$L_DPC         00000080     D
UCB$L_DUETIM      0000005C     D
UCB$L_DX_BFPNT    0000009C     D
UCB$L_DX_BUF      00000098     D
UCB$L_DX_RXDB     000000A0     D
UCB$L_EMB         00000078     D
UCB$L_FIRST       00000014     D
UCB$L_FPC         0000000C     D
UCB$L_FQBL        00000004     D
UCB$L_FQFL        00000000     D
UCB$L_FR3         00000010     D
UCB$L_FR4         00000014     D
UCB$L_IOQBL       00000044     D
UCB$L_IOQFL       00000040     D
UCB$L_IRP         0000004C     D
UCB$L_LINK        0000002C     D
UCB$L_LOGADR      00000064     D
UCB$L_MAXBLOCK    00000084     D
UCB$L_MB_MBX      0000007C     D
UCB$L_MB_PORT     0000008C     D
UCB$L_MB_RAST     00000078     D
UCB$L_MB_SHB      00000080     D
UCB$L_MB_WAST     00000074     D
UCB$L_MB_WIOQBL   00000088     D
UCB$L_MB_WIOQFL   00000084     D
UCB$L_MEDIA       0000008C     D
UCB$L_NT_DATSSB   00000074     D
UCB$L_NT_INTSSB   00000078     D
UCB$L_OPCNT       00000060     D
UCB$L_OWNUIC      0000001C     D
UCB$L_PID         00000028     D
UCB$L_RQBL        00000004     D
UCB$L_RQFL        00000000     D
UCB$L_SVAPTE      00000068     D
UCB$L_SVPN        00000064     D
UCB$L_TT_DECHAR   000000A8     D
UCB$L_TT_RDUE     0000008C     D
UCB$L_TT_RTIMOU   000000B8     D
UCB$L_VCB         00000030     D
UCB$M_BSY       = 00000100     D
UCB$M_CANCEL    = 00000008     D
UCB$M_INT       = 00000002     D
UCB$M_TIM       = 00000001     D
UCB$M_TIMOUT    = 00000040     D
UCB$T_PARTNER     0000000C     D
UCB$V_BSY       = 00000008     D
UCB$V_ERLOGIP   = 00000002     D
UCB$W_BCNT        0000006E     D
UCB$W_BCR         00000096     D
UCB$W_BOFF        0000006C     D
UCB$W_BUFQUO      00000018     D
UCB$W_BYTESTOGO   0000003E     D
UCB$W_CHARGE      0000004A     D
UCB$W_CYLINDERS   0000003E     D
UCB$W_DA          0000008C     D
UCB$W_DC          0000008E     D
UCB$W_DEVBUFSIZ   0000003A     D
UCB$W_DEVSTS      0000005A     D
UCB$W_DIRSEQ      00000088     D
UCB$W_DSTADDR     00000018     D
UCB$W_DX_BCR      000000A4     D
UCB$W_ECT         00000090     D
UCB$W_EC2         00000092     D
UCB$W_ERRCNT      00000072     D
UCB$W_FUNC        0000007E     D
UCB$W_MB_SEED     00000000     D
UCB$W_MSGCNT      00000016     D
UCB$W_MSGMAX      00000014     D
UCB$W_NT_CHAN     0000007C     D
UCB$W_OFFSET      000000BA     D
UCB$W_REFC        00000050     D
UCB$W_SIZE        00000008     D
UCB$W_SRCADDR     0000001A     D
UCB$W_STS         00000058     D
UCB$W_TT_DESIZE   000000A5     D
UCB$W_UNIT        00000048     D
UCB$W_VPROT       0000001A     D
VEC$B_DATAPATH    00000013     D
VEC$B_NUMREG      00000012     D
VEC$C_LENGTH      00000024     D
VEC$K_LENGTH      00000024     D
VEC$L_ADP         00000014     D
VEC$L_IDB         00000008     D
VEC$L_INITIAL     0000000C     D
VEC$L_START       0000001C     D
VEC$L_UNITDISC    00000020     D
VEC$L_UNITINIT    00000018     D
VEC$Q_DISPATCH    00000000     D
VEC$S_DATAPATH  = 00000005     D
VEC$V_DATAPATH  = 00000000     D
VEC$V_MAPLOCK   = 0000000F     D
VEC$V_PATHLOCK  = 00000007     D
VEC$W_MAPREG      00000010     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	000002FF (767.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ADP$C_NUMDATAP	=0000C010		#-500 (1)
ADP$L_DPQBL	00000018		509 (1)
ADP$L_DPQFL	00000014		446 (1)
ADP$L_MRQBL	00000020		601 (1)
ADP$L_MRQFL	0000001C		558 (1) 565 (1)
ADP$W_DPBITMAP	00000024		458 (1) 500 (1) 505 (1)
ADP$W_MRBITMAP	00000026		630 (1) 634 (1) 674 (1) 678 (1) 681 (1)
BUG$CHECK	0000C000-XR		459 (1)
CRB$B_MASK	0000000E		246 (1) #-261 (1) 311 (1)
CRB$L_INTD	00000014		#-247 (1) #-310 (1) #-438 (1) #-439 (1) 443 (1)
			453 (1) 498 (1) #-499 (1) 504 (1) 551 (1)
			#-553 (1) #-555 (1) #-627 (1) #-667 (1) 668 (1)
			#-669 (1) #-677 (1) #-683 (1)
CRB$L_LINK	00000010		#-238 (1) #-242 (1) #-297 (1) #-301 (1)
CRB$L_WQBL	00000004		#-309 (1)
CRB$L_WQFL	00000000		250 (1) 317 (1)
CRB$M_BSY	=00000001		#-261 (1)
CRB$V_BSY	=00000000		#-246 (1) #-311 (1)
DDB$L_DDT	0000000C		#-203 (1) #-408 (1)
DDT$L_REGDUMP	00000010		#-203.1 (1)
DDT$L_START	00000000		#-408.1 (1)
EMB$B_DV_ERTCNT	=00000010		#-363 (1)
EMB$Q_DV_IOSB	=00000012		#-364 (1)
EMB$W_DV_STS	=0000001A		#-362 (1)
ERL$RELEASEMB	00000000-XR		#-365 (1)
EXE$GL_ABSTIM	00000000-XR		#-737.1 (1) #-771.1 (1)
EXE$GQ_SYSTIME	00000000-XR		#-200 (1) #-406 (1)
IDB$L_CSR	00000000		#-255 (1) #-312 (1)
IDB$L_OWNER	00000004		#-248 (1) #-256 (1) #-260 (1) #-313 (1) #-318 (1)
IOC$ALLOSPT	000002F4-R	802 (1)	
IOC$ALOUBAMAP	0000024B-R	659 (1)	#-560 (1) #-597 (1)
IOC$ALOUBAMAPN	00000247-R	656 (1)	
IOC$ALTUBAMAP	00000229-R	626 (1)	#-557 (1) #-684 (1)
IOC$CANCELIO	00000000-R	96 (1)	
IOC$DIAGBUFILL	00000017-R	195 (1)	
IOC$GL_PSBL	00000000-XR		356.1 (1)
IOC$INITIATE	0000011D-R	393 (1)	#-367 (1)
IOC$RELCHAN	00000052-R	240 (1)	#-370 (1) #-775 (1)
IOC$RELDATAP	00000150-R	436 (1)	
IOC$RELMAPREG	000001DD-R	549 (1)	
IOC$RELSCHAN	00000048-R	236 (1)	
IOC$REQCOM	000000DD-R	346 (1)	
IOC$REQDATAP	000001A4-R	495 (1)	
IOC$REQDATAPNW	000001A0-R	492 (1)	
IOC$REQMAPREG	00000217-R	596 (1)	
IOC$REQPCHANH	000000A9-R	303 (1)	
IOC$REQPCHANL	000000B2-R	307 (1)	
IOC$REQSCHANH	00000095-R	295 (1)	
IOC$REQSCHANL	0000009F-R	299 (1)	
IOC$RETURN	000002AD-R	706 (1)	

```
IOC$WFIKPCH     000002AE-R  733     (1)
IOC$WFIRLCH     000002D0-R  767     (1)
IPL$_IOPOST     =00000004                   #-359    (1)
IRP$L_DIAGBUF   00000044                    #-198    (1)    #-405    (1)
IRP$L_MEDIA     00000034                    #-348    (1)
IRP$L_PID       0000000C                    #-98     (1)
IRP$L_SVAPTE    0000002C                    #-402    (1)
IRP$V_DIAGBUF   =00000007                   #-197    (1)    #-404    (1)
IRP$W_CHAN      00000028                    #-100    (1)
IRP$W_STS       0000002A                      197    (1)      404    (1)
MMG$GL_SPTBASE  00000000-XR                 #-802.1  (1)
PCB$L_PID       00000060                    #-98     (1)
PR$_IPL         =00000012                   #-740    (1)    #-774    (1)
PR$_SIRR        =00000014                   #-359    (1)
RELEASE         0000011A-R  369     (1)     #-319    (1)
SS$_NORMAL      00000000-XR                 #-511    (1)
UCB$B_ERTCNT    00000070                    #-201    (1)    #-363    (1)
UCB$L_CRB       00000020                    #-237    (1)    #-241    (1)    #-245    (1)    #-296    (1)    #-300    (1)
                                            #-304    (1)    #-308    (1)    #-437    (1)    #-450    (1)    #-497    (1)
                                            #-550    (1)    #-666    (1)
UCB$L_DDB       00000024                    #-202    (1)    #-407    (1)
UCB$L_DUETIM    0000005C                    #-737.1  (1)    #-771.1  (1)
UCB$L_EMB       00000078                    #-361    (1)
UCB$L_FPC       0000000C                      257    (1)    #-316    (1)      455    (1)    #-508    (1)      563    (1)
                                            #-600    (1)    #-736    (1)    #-770    (1)
UCB$L_FQFL      00000000                      317    (1)      509    (1)      565    (1)      601    (1)
UCB$L_FR3       00000010                    #-254    (1)    #-315    (1)    #-454    (1)    #-507    (1)    #-562    (1)
                                            #-599    (1)    #-735    (1)    #-769    (1)
UCB$L_IOQFL     00000040                      366    (1)
UCB$L_IRP       0000004C                    #-196    (1)    #-347    (1)    #-394    (1)
UCB$L_OPCNT     00000060                    #-349    (1)
UCB$L_SVAPTE    00000068                    #-402    (1)
UCB$M_BSY       =00000100                   #-368    (1)
UCB$M_CANCEL    =00000008                   #-102    (1)    #-403    (1)
UCB$M_INT       =00000002                   #-737    (1)    #-771    (1)
UCB$M_TIM       =00000001                   #-737    (1)    #-771    (1)
UCB$M_TIMOUT    =00000040                   #-403    (1)    #-739    (1)    #-773    (1)
UCB$V_BSY       =00000008                   #-97     (1)
UCB$V_ERLOGIP   =00000002                   #-360    (1)
UCB$W_BCNT      0000006E                    #-661    (1)
UCB$W_BOFF      0000006C                    #-662    (1)
UCB$W_STS       00000058                    #-102    (1)      360    (1)    #-362    (1)    #-368    (1)    #-403    (1)
                                            #-737    (1)    #-739    (1)    #-771    (1)    #-773    (1)      97    (1)
VEC$B_DATAPATH  00000013                    #-439    (1)      443    (1)      453    (1)      498    (1)      504    (1)
VEC$B_NUMREG    00000012                    #-627    (1)    #-669    (1)
VEC$L_ADP       00000014                    #-438    (1)    #-499    (1)    #-553    (1)    #-667    (1)
VEC$L_IDB       00000008                    #-247    (1)    #-310    (1)
VEC$S_DATAPATH  =00000005                   #-442    (1)    #-445    (1)    #-452    (1)    #-503    (1)
VEC$V_DATAPATH  =00000000                   #-441    (1)    #-444    (1)    #-451    (1)    #-502    (1)
VEC$V_MAPLOCK   =0000000F                   #-551    (1)    #-668    (1)
VEC$V_PATHLOCK  =00000007                   #-498    (1)
VEC$W_MAPREG    00000010                      551    (1)    #-555    (1)      668    (1)    #-677    (1)    #-683    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ADPDEF	2	55 (1)	55 (1)
$CADEF	1	56 (1)	56 (1)
$CRBDEF	1	57 (1)	57 (1)
$DDBDEF	1	58 (1)	58 (1)
$DDTDEF	1	59 (1)	59 (1)
$DEFINI	1	55 (1)	55 (1) 56 (1) 57 (1) 58 (1) 59 (1)
			60 (1) 61 (1) 62 (1) 63 (1) 64 (1)
			65 (1) 66 (1) 67 (1) [illegible] (1) 69 (1)
$EMBDEF	1	60 (1)	60 (1)
$EMBDVDEF	2	60 (1)	60 (1)
$EMBETDEF	3	60 (1)	60 (1)
$EMBHDDEF	1	60 (1)	60 (1)
$EMBTSDEF	1	60 (1)	60 (1)
$IDBDEF	1	61 (1)	61 (1)
$IPLDEF	1	62 (1)	62 (1)
$IRPDEF	4	63 (1)	63 (1)
$JIBDEF	3	64 (1)	64 (1)
$LOGDEF	1	65 (1)	65 (1)
$PCBDEF	4	66 (1)	66 (1)
$PRDEF	4	67 (1)	67 (1)
$UCBDEF	10	68 (1)	68 (1)
$VECDEF	2	69 (1)	69 (1)
BUG_CHECK	1	459 (1)	459 (1)
ENBINT	1	740 (1)	740 (1) 774 (1)
SOFTINT	1	359 (1)	359 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.11	00:00:00.25
Command processing	135	00:00:00.79	00:00:01.95
Pass 1	588	00:00:14.80	00:00:24.86
Symbol table sort	0	00:00:01.03	00:00:02.35
Pass 2	182	00:00:03.40	00:00:06.21
Symbo. table output	28	00:00:00.22	00:00:00.48
Psect synopsis output	6	00:00:00.02	00:00:00.02
Cross-reference output	41	00:00:00.48	00:00:00.49
Assembler run totals	1016	00:00:20.86	00:00:36.61

The working set limit was 1000 pages.
61035 bytes (120 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 712 non-local and 35 local symbols.
739 source lines were read in Pass 1, producing 0 object records in Pass 2.
104 pages of virtual memory were used to define 31 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	10
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	10
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	27

1114 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) IOSNPG/UPDA=(IOSNPG.UPD,IOSNPG.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 148 CONVERT DEVICE NAME AND UNIT
(1) 186 FIND FREE I/O CHANNEL
(1) 226 SEARCH FOR DEVICE
(1) 383 UNLOCK I/O DATA BASE AND RETURN STATUS
(1) 409 VERIFY I/O CHANNEL NUMBER

```
V54                     0000     .1          .TITLE  IOSPGD  *** IOSPGD services for QIO
V54                     0000     .2          .IDENT  /05-02/
V54                     0000     .3          .LIST   MEB
V54                     0000     .4          .NLIST  CND
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                        0000     11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER     *
                        0000     13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
                        0000     15 ;*  TRANSFERRED.                                                             *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
                        0000     18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                        0000     19 ;*  CORPORATION.                                                             *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 13-JUN-76
                        0000     27 ;
                        0000     28 ; PAGED I/O RELATED SUBROUTINES
                        0000     29 ;
                        0000     30 ;
V54                     0000     .1 ;
V54                     0000     .2 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
V54                     0000     .3 ;
V54                     0000     .4 ;               Dave Butenhof   14-may-1980, Version 5.4
V54                     0000     .5 ;       02      Modify VMS V2.0 source for Supervisor environment
                        0000     31 ;       V04     LMK0001         LEN KAWELL      27-JUL-1979
                        0000     32 ;               CHANGE INTERFACE TO IOC$CREATE_UCB TO ASSUME THAT THE
                        0000     33 ;               I/O DATABASE IS LOCKED FOR WRITE ACCESS.
                        0000     34 ;
                        0000     35 ;       V03     SPR23081        LEN KAWELL      28-MAR-1979
                        0000     36 ;               CHANGE OVERFLOW CHECK IN IOC$SEARCHDEV SO THAT UNIT
                        0000     37 ;               NUMBERS BETWEEN 32768 AND 65535 ARE VALID.
                        0000     38 ;
                        0000     39 ;       V02     SPR20488                LEN KAWELL      7-FEB-1979
                        0000     40 ;               RESTORE PCB ADDRESS WHEN CHECKING FOR ALLSPOOL
                        0000     41 ;               PRIVILEGE DURING GENERIC ALLOCATION OF A SPOOLED DEVICE.
                        0000     42 ;
                        0000     43 ; MACRO LIBRARY CALLS
                        0000     44 ;
V54                     0000     .4
-1                      0000     46          $ACBDEF                         ;DEFINE AST CONTROL BLOCK
                        0000     47          $CCBDEF                         ;DEFINE CCB OFFSETS
                        0000     48          $CRBDEF                         ;DEFINE CRB OFFSETS
                        0000     49          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     50          $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
```

```
                                0000     51          $IPLDEF                         ;DEFINE IPL LEVELS
                                0000     52          $LOGDEF                         ;DEFINE LOG OFFSETS
                                0000     53          $PCBDEF                         ;DEFINE PCB OFFSETS
                                0000     54          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                                0000     55          $PRVDEF                         ;DEFINE PRIVILEGE BITS
                                0000     56          $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                                0000     57          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                                0000     58          $UCBDEF                         ;DEFINE UCB OFFSETS
                                0000     59
                                0000     60 ;
                                0000     61 ; LOCAL SYMBOLS
                                0000     62 ;
                                0000     63 ; CHARACTER DEFINITIONS
                                0000     64 ;
                                0000     65
                     0000003A   0000     66 COLON=58                                ;COLON
V54                            0000     .1          .PSECT  SEP, SHR, EXE, WRT, LONG
                     00000039   0000     67 NINE=57                                 ;DIGIT 9
                     00000041   0000     68 UCA=65                                  ;UPPER CASE A
                     00000064   0000     69 UCZ=100                                 ;UPPER CASE Z
                     0000005F   0000     70 UNDERSCORE=95                           ;UNDERSCORE
                     00000030   0000     71 ZERO=48                                 ;DIGIT 0
                                0000     72
                                0000     73 ;
                                0000     74 ; LOCAL DATA
                                0000     75 ;
                                0000     76
-1                 21 43 41 21 5F 00'  0000     78 DEVCTL: .ASCIC  /_!AC!UW:/              ;DEVICE NAME CONVERSION CONTROL STRING
                      3A 57 55  0006
                            08  0000
```

```
                                0009   148          .SBTTL  CONVERT DEVICE NAME AND UNIT
                                0009   149 ;+
                                0009   150 ; IOC$CVT_DEVNAM - CONVERT DEVICE NAME AND UNIT
                                0009   151 ;
                                0009   152 ; THIS ROUTINE IS CALLED TO CONVERT A DEVICE NAME AND UNIT NUMBER TO A PHYSICAL
                                0009   153 ; DEVICE NAME STRING.
                                0009   154 ;
                                0009   155 ; INPUTS:
                                0009   156 ;
                                0009   157 ;       R0 = LENGTH OF OUTPUT BUFFER.
                                0009   158 ;       R1 = ADDRESS OF OUTPUT BUFFER.
                                0009   159 ;       R6 = ADDRESS OF DEVICE UCB.
                                0009   160 ;
                                0009   161 ; OUTPUTS:
                                0009   162 ;
                                0009   163 ;       THE DEVICE NAME AND UNIT NUMBER ARE CONVERTED AND STORED IN THE SPECIFIED
                                0009   164 ;       OUTPUT BUFFER. THE FOLLOWING REGISTER VALUES ARE RETURNED:
                                0009   165 ;
                                0009   166 ;               R0 = FINAL CONVERSION STATUS.
                                0009   167 ;               R1 = LENGTH OF CONVERSION STRING.
                                0009   168 ;
                                0009   169 ;       R3 IS PRESERVED ACROSS CALL.
                                0009   170 ;-
                                0009   171
                                0009   172 IOC$CVT_DEVNAM::                          ;CONVERT DEVICE NAME AND UNIT
      7E    50    7D            0009   173          MOVQ    R0,-(SP)                ;SAVE OUTPUT BUFFER DESCRIPTOR
      52    5E    D0            000C   174          MOVL    SP,R2                   ;SET ADDRESS OF OUTPUT BUFFER DESCRIPTOR
51    EE    AF    9E            000F   175          MOVAB   DEVCTL,R1               ;GET ADDRESS OF CONVERSION CONTROL STRING
      50    81    9A            0013   176          MOVZBL  (R1)+,R0                ;GET LENGTH OF CONVERSION CONTROL STRING
      7E    50    7D            0016   177          MOVQ    R0,-(SP)                ;SAVE CONVERSION CONTROL STRING DESCRIPTOR
      51    5E    D0            0019   178          MOVL    SP,R1                   ;SET ADDRESS OF CONTROL STRING DESCRIPTOR
50     24   A6    D0            001C   179          MOVL    UCB$L_DDB(R6),R0        ;GET ADDRESS OF DDB
50     14   A0    9E            0020   180          MOVAB   DDB$T_NAME(R0),R0       ;GET ADDRESS OF DEVICE NAME
                                0024   181          $FAO_S  (R1),(R1),(R2),R0,UCB$W_UNIT(R6) ;CONVERT DEVICE AND UNIT
      48    A6    DD            0024                 PUSHL   UCB$W_UNIT(R6)
            50    DD            0027                 PUSHL   R0
            62    7F            0029                 PUSHAQ  (R2)
            61    3F            002B                 PUSHAW  (R1)
            61    7F            002D                 PUSHAQ  (R1)
            05    FB            002F                 CALLS   #$$T2,G^SYS$FAO
00000000'GF                     0031
      51    6E    3C            0036   182          MOVZWL  (SP),R1                 ;GET LENGTH OF OUTPUT STRING
      5E    10    C0            0039   183          ADDL    #16,SP                  ;REMOVE DESCRIPTORS FROM STACK
                  05            003C   184          RSB                             ;
```

```
                                003D   186          .SBTTL  FIND FREE I/O CHANNEL
                                003D   187 ;+
                                003D   188 ; IOC$FFCHAN - FIND FREE I/O CHANNEL
                                003D   189 ;
                                003D   190 ; THIS ROUTINE IS CALLED TO SEARCH THE I/O CHANNEL TABLE FOR A FREE CHANNEL.
                                003D   191 ;
                                003D   192 ; INPUTS:
                                003D   193 ;
                                003D   194 ;       NONE.
                                003D   195 ;
                                003D   196 ; OUTPUTS:
                                003D   197 ;
                                003D   198 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FIND FREE I/O CHANNEL.
                                003D   199 ;
                                003D   200 ;               R0 = SS$_NOIOCHAN - NO I/O CHANNEL AVAILABLE.
                                003D   201 ;
                                003D   202 ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                                003D   203 ;
                                003D   204 ;               R1 = AVAILABLE CHANNEL NUMBER.
                                003D   205 ;
                                003D   206 ;       R3 IS PRESERVED ACROSS CALL.
                                003D   207 ;-
                                003D   208
                                003D   209 IOC$FFCHAN::                                 ;FIND FREE I/O CHANNEL
00000000'EF   C1                003D   210          ADDL3   CTL$GL_CCBBASE,#CCB$B_AMOD,R0   ;BASE AND OFFSET TO TEST ASSIGNMENT
     50   09                    0043
     51   10   CE                0045   211          MNEGL   #CCB$C_LENGTH,R1        ;SET STARTING CHANNEL INDEX
00000000'9F   3C                0048   212          MOVZWL  @#CTL$GW_NMIOCH,R2      ;GET NUMBER OF I/O CHANNELS
          52                    004E
        6041   95                004F   213 10$:     TSTB    (R0)[R1]                ;CHANNEL ASSIGNED?
          0C   13                0052   214          BEQL    20$                     ;IF EQL NO
     51   10   C2                0054   215          SUBL    #CCB$C_LENGTH,R1        ;CALCULATE NEXT CHANNEL INDEX
       F5 52   F5                0057   216          SOBGTR  R2,10$                  ;ANY MORE CCB'S TO EXAMINE?
50   01B4 8F   3C                005A   217          MOVZWL  #SS$_NOIOCHAN,R0        ;INDICATE FAILURE
                05                005F   218          RSB                             ;
     50   01   3C                0060   219 20$:     MOVZWL  #SS$_NORMAL,R0          ;INDICATE SUCCESS
                05                0063   220          RSB                             ;
```

```
                     0064   222          .SBTTL  SEARCH FOR DEVICE
                     0064   223 ;+
                     0064   224 ; IOC$SEARCHDEV - SEARCH FOR PHYSICAL DEVICE
                     0064   225 ; IOC$SEARCHGEN - SEARCH FOR GENERIC DEVICE
                     0064   226 ;
                     0064   227 ; THIS ROUTINE IS CALLED TO SEARCH THE DEVICE DATA BASE FOR A SPECIFIED
                     0064   228 ; DEVICE. IT IS ASSUMED THAT THE DEVICE DATA BASE HAS BEEN LOCKED FOR
                     0064   229 ; READ ACCESS.
                     0064   230 ;
                     0064   231 ; INPUTS:
                     0064   232 ;
                     0064   233 ;       R1 = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
                     0064   234 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0064   235 ;
                     0064   236 ; OUTPUTS:
                     0064   237 ;
                     0064   238 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FIND DEVICE.
                     0064   239 ;
                     0064   240 ;               R0 = SS$_IVDEVNAM - INVALID DEVICE NAME.
                     0064   241 ;               R0 = SS$_NONLOCAL - NONLOCAL DEVICE.
                     0064   242 ;               R0 = SS$_NOSUCHDEV - NO SUCH DEVICE.
                     0064   243 ;
                     0064   244 ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                     0064   245 ;
                     0064   246 ;               R1 = UCB ADDRESS OF DEVICE UNIT.
                     0064   247 ;-
                     0064   248
                     0064   249          .ENABL  LSB
                     0064   250 IOC$SEARCHDEV::                              ;SEARCH FOR PHYSICAL DEVICE
          52   02    D0     0064   251          MOVL    #2,R2                 ;INDICATE PHYSICAL DEVICE SEARCH
               03    11     0067   252          BRB     5$                    ;
                     0069   253 IOC$SEARCHGEN::                             ;SEARCH FOR GENERIC DEVICE
          52   01    D0     0069   254          MOVL    #1,R2                 ;INDICATE GENERIC DEVICE SEARCH
          01F4 8F    BB     006C   255 5$:      PUSHR   #^M<R2,R4,R5,R6,R7,R8> ;SAVE REGISTERS
     52     40 8F    9A     0070   256          MOVZBL  #LOG$C_NAMLENGTH,R2   ;SET MAXIMUM LENGTH OF RESULT STRING
          5E   52    C2     0074   257          SUBL    R2,SP                 ;ALLOCATE SPACE FOR RESULT STRING
V54       54   61    3C     0077    .1          MOVZWL  (R1),R4               ;GET LENGTH OF NAME STRING IN BYTES
V54            69    13     007A    .2          BEQL    70$                   ;IF EQL INVALID DEVICE NAME
V54       55   04    A1     D0     007C    .3   MOVL    4(R1),R5              ;GET ADDRESS OF NAME STRING
-7        85   5F    8F     91     0080   265   CMPB    #UNDERSCORE,(R5)+     ;DEVICE NAME START WITH UNDERSCORE?
               06    12     0084   266          BNEQ    7$                    ;IF NEQ NO
               54    D7     0086   267          DECL    R4                    ;REDUCE LENGTH OF DEVICE NAME
               5B    13     0088   268          BEQL    70$                   ;IF EQL INVALID DEVICE NAME
               10    11     008A   269          BRB     10$                   ;
     7S        54    3A     3A     008C   270 7$:      LOCC    #^A/:/,R4,-(R5)       ;SEARCH STRING FOR A COLON
               0A    13     0090   271          BEQL    10$                   ;IF EQL COLON NOT FOUND
               50    D7     0092   272          DECL    R0                    ;POSSIBLY A NODE NAME?
               06    13     0094   273          BEQL    10$                   ;IF EQL NO
          01   A1    3A     91     0096   274          CMPB    #^A/:/,1(R1)          ;NEXT CHARACTER A COLON?
               42    13     009A   275          BEQL    50$                   ;IF EQL YES
          50        54    7D     009C   276 10$:     MOVQ    R4,R0                 ;COPY DEVICE NAME PARAMETERS
     61     64 8F    91     009F   277 20$:     CMPB    #UCZ,(R1)             ;POSSIBLY UPPER CASE ALPHABETIC?
               40    1F     00A3   278          BLSSU   70$                   ;IF LSSU NO
     61     41 8F    91     00A5   279          CMPB    #UCA,(R1)             ;UPPER CASE ALPHABETIC?
               05    1A     00A9   280          BGTRU   30$                   ;IF GTRU NO
               51    D6     00AB   281          INCL    R1                    ;POINT TO NEXT CHARACTER
               EF 50 F5     00AD   282          SOBGTR  R0,20$                ;ANY MORE CHARACTERS TO SCAN?
```

```
                56   D4   00B0    283 30$:    CLRL    R6                              ;CLEAR UNIT NUMBER
              54 50   C2   00B2    284         SUBL    R0,R4                           ;CALCULATE LENGTH OF DEVICE NAME
                 2E   13   00B5    285         BEQL    70$                             ;IF EQL NO DEVICE NAME SPECIFIED
                 50   D7   00B7    286 40$:    DECL    R0                              ;ANY MORE CHARACTERS IN STRING?
                 31   19   00B9    287         BLSS    80$                             ;IF LSS NO
              52 81   9A   00BB    288         MOVZBL  (R1)+,R2                        ;GET NEXT CHARACTER
              52 3A   91   00BE    289         CMPB    #COLON,R2                       ;DEVICE NAME TERMINATOR?
                 29   13   00C1    290         BEQL    80$                             ;IF EQL YES
           40 AE 02   C8   00C3    291         BISL    #2,LOG$C_NAMLENGTH(SP)          ;SET EXPLICIT UNIT NUMBER FLAG
              52 30   82   00C7    292         SUBB    #ZERO,R2                        ;POSSIBLY A DECIMAL DIGIT?
                 19   19   00CA    293         BLSS    70$                             ;IF LSS NO
              52 09   91   00CC    294         CMPB    #NINE-ZERO,R2                   ;DECIMAL DIGIT?
                 14   19   00CF    295         BLSS    70$                             ;IF LSS NO
              56 05   A4   00D1    296         MULW    #5,R6                           ;SCALE CURRENT UNIT NUMBER BY 10
                           00D4    297                                                 ; IN TWO STEPS, FOR OVERFLOW CHECK
                 0F   1D   00D4    298         BVS     70$                             ;IF VS OVERFLOW - NUMBER TOO BIG
              56 56   A0   00D6    299         ADDW    R6,R6                           ;SECONL STEP OF SCALE BY 10
              56 52   C0   00D9    300         ADDL    R2,R6                           ;ADD NEW DIGIT TO ACCUMULATION
                 D9   11   00DC    301         BRB     40$                             ;
                           00DE    302
                           00DE    303 ;
                           00DE    304 ; NONLOCAL DEVICE
                           00DE    305 ;
                           00DE    306
        50 08F0 8F   3C   00DE    307 50$:    MOVZWL  #SS$_NONLOCAL,R0                ;SET NONLOCAL DEVICE
                 35   11   00E3    308 60$:    BRB     120$                            ;
                           00E5    309
                           00E5    310 ;
                           00E5    311 ; INVALID DEVICE NAME
                           00E5    312 ;
                           00E5    313
        50 0144 8F   3C   00E5    314 70$:    MOVZWL  #SS$_IVDEVNAM,R0                ;SET INVALID DEVICE NAME
                 2E   11   00EA    315         BRB     120$                            ;
                           00EC    316
                           00EC    317 ;
                           00EC    318 ; SEARCH DEVICE DATA BASE FOR NAME/UNIT MATCH
                           00EC    319 ;
                           00EC    320
           57   01   D0   00EC    321 80$:    MOVL    #1,R7                           ;SET DEVICE NAME LENGTH ADJUSTMENT VALUE
 00000000'EF    DE   00EF    322         MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK,R8 ;GET ADDRESS OF I/O DATABASE LISTHEAD
              58           00F5
                 28   10   00F6    323 90$:    BSBB    SEARCHDEV                       ;SEARCH FOR DEVICE NAME MATCH
                 09   12   00F8    324         BNEQ    100$                            ;IF NEQ MATCH NOT FOUND
-3               43   10   00FA    328 95$:    BSBB    SEARCHUNIT                      ;SEARCH FOR UNIT NUMBER MATCH
           1B 50   E8   00FC    329         BLBS    R0,120$                         ;IF LBS UNIT NUMBER MATCH FOUND
        F3 40 AE   E8   00FF    330         BLBS    LOG$C_NAMLENGTH(SP),90$ ;IF LBS GENERIC DEVICE NAME SEARCH
              57   D4   0103    331 100$:   CLRL    R7                              ;CLEAR DEVICE NAME LENGTH ADJUSTMENT VALUE
 00000000'EF    DE   0105    332         MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK,R8 ;GET ADDRESS OF I/O DATABASE LISTHEAD
              58           010B
                 15   10   010C    333         BSBB    SEARCHDEV                       ;SEARCH FOR DEVICE NAME MATCH
                 05   12   010E    334         BNEQ    110$                            ;IF NEQ MATCH NOT FOUND
                 2D   10   0110    335         BSBB    SEARCHUNIT                      ;SEARCH FOR UNIT NUMBER MATCH
           05 50   E8   0112    336         BLBS    R0,120$                         ;IF LBS UNIT NUMBER MATCH FOUND
        50 0908 8F   3C   0115    337 110$:   MOVZWL  #SS$_NOSUCHDEV,R0               ;SET NO SUCH DEVICE
           5E   40 AE   9E   011A    338 120$:   MOVAB   LOG$C_NAMLENGTH(SP),SP  ;REMOVE DEVICE NAME STRING FROM STACK
              01F4 8F   BA   011E    339         POPR    #^M<R2,R4,R5,R6,R7,R8>  ;RESTORE REGISTERS
                 05   0122    340         RSB                                     ;
```

```
                                0123    341          .DSABL  LSB
                                0123    342
                                0123    343 ;
                                0123    344 ; SUBROUTINE TO SEARCH FOR DEVICE NAME MATCH
                                0123    345 ;
                                0123    346
                                0123    347 SEARCHDEV:                                  ;SEARCH FOR DEVICE NAME
          58    68    D0      0123    348 10$:     MOVL    DDB$L_LINK(R8),R8          ;GET ADDRESS OF NEXT DDB
                14    13      0126    349          BEQL    20$                        ;IF EQL END OF LIST
     50     14    A8    DE    0128    350          MOVAL   DDB$T_NAME(R8),R0          ;GET ADDRESS OF GENERIC DEVICE NAME
51   80          57    83    012C    351          SUBB3   R7,(R0)+,R1                ;CALCULATE LENGTH OF STRING TO COMPARE
          54    51    91      0130    352          CMPB    R1,R4                      ;LENGTH OF NAMES MATCH?
                EE    12      0133    353          BNEQ    10$                        ;IF NEQ NO
65   60         54    29      0135    354          CMPC    R4,(R0),(R5)               ;COMPARE DEVICE NAMES
                E8    12      0139    355          BNEQ    10$                        ;IF NEQ NAMES DO NOT MATCH
                      05      013B    356          RSB                                ;
                58    D6      013C    357 20$:     INCL    R8                         ;INDICATE SEARCH FAILURE
                      05      013E    358          RSB                                ;
                              013F    359
                              013F    360 ;
                              013F    361 ; SUBROUTINE TO SEARCH FOR UNIT NUMBER MATCH
                              013F    362 ;
                              013F    363
                              013F    364 SEARCHUNIT:                                 ;SEARCH FOR UNIT NUMBER
                50    D4      013F    365          CLRL    R0                         ;ASSUME SEARCH FAILURE
     51   D8    A8    DE    0141    366          MOVAL   DDB$L_UCB-UCB$L_LINK(R8),R1 ;GET ADDRESS OF NEXT UCB ADDRESS
     51   2C    A1    D0    0145    367 10$:     MOVL    UCB$L_LINK(R1),R1          ;GET ADDRESS OF NEXT UCB
                0F    13      0149    368          BEQL    40$                        ;IF EQL END OF LIST
08 44 AE         01    E1    014B    369          BBC     #1,LOG$C_NAMLENGTH+4(SP),20$ ;IF CLR, GENERIC SEARCH
   48 A1         56    B1    0150    370          CMPW    R6,UCB$W_UNIT(R1)          ;UNIT NUMBER MATCH?
                02    13      0154    371          BEQL    30$                        ;IF EQL YES
                ED    11      0156    372          BRB     10$                        ;
V54                             0158      ,1 20$:
-7              50    D6      0158    380 30$:     INCL    R0                         ;INDICATE UNIT NUMBER MATCH
                      05      015A    381 40$:     RSB                                ;
```

```
                                015B   383          .SBTTL  UNLOCK I/O DATA BASE AND RETURN STATUS
                                015B   384 ;+
                                015B   385 ; IOC$UNLOCK - UNLOCK I/O DATA BASE AND RETURN STATUS
                                015B   386 ;
                                015B   387 ; THIS ROUTINE IS JUMPED TO AT THE END OF AN I/O RELATED SYSTEM SERVICE TO
                                015B   388 ; UNLOCK THE I/O DATA BASE, SET THE CURRENT PROCESSOR PRIORITY TO ZERO,
                                015B   389 ; AND TO RETURN STATUS TO THE CHANGE MODE DISPATCHER.
                                015B   390 ;
                                015B   391 ; INPUTS:
                                015B   392 ;
                                015B   393 ;       R0 = FINAL SYSTEM SERVICE STATUS VALUE.
                                015B   394 ;
                                015B   395 ; OUTPUTS:
                                015B   396 ;
                                015B   397 ;       THE I/O DATA BASE IS UNLOCKED, THE CURRENT PROCESSOR PRIORITY IS SET
                                015B   398 ;       TO ZERO, AND A RETURN TO THE CHANGE MODE DISPATCHER IS EXECUTED.
                                015B   399 ;-
                                015B   400
                                015B   401 IOC$UNLOCK::                                ;UNLOCK I/O DATA BASE AND RETURN STATUS
V54                             015B    .1          SETIPL  #0                          ;ALLOW ALL INTERRUPTS
                          12    015B                MTPR    #0,S^#PR$_IPL
-5                        04    015E   407          RET                                 ;
```

```
                                015F   409          .SBTTL  VERIFY I/O CHANNEL NUMBER
                                015F   410 ;+
                                015F   411 ; IOC$VERIFYCHAN - VERIFY I/O CHANNEL NUMBER
                                015F   412 ;
                                015F   413 ; THIS ROUTINE IS CALLED TO VERIFY AND TRANSLATE AN I/O CHANNEL NUMBER TO
                                015F   414 ; A CCB ADDRESS. THE CHANNEL IS CHECKED FOR ACCESSIBILITY BY THE PREVIOUS
                                015F   415 ; ACCESS MODE.
                                015F   416 ;
                                015F   417 ; INPUTS:
                                015F   418 ;
                                015F   419 ;       R0 = I/O CHANNEL NUMBER.
                                015F   420 ;
                                015F   421 ; OUTPUTS:
                                015F   422 ;
                                C15F   423 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO VERIFY.
                                015F   424 ;
                                015F   425 ;               R0 = SS$_IVCHAN - INVALID CHANNEL NUMBER.
                                015F   426 ;               R0 = SS$_NOPRIV - NO PRIVILEGE TO ACCESS CHANNEL.
                                015F   427 ;               R1 = ADDRESS OF CCB IF R0 = SS$_NOPRIV
                                015F   428 ;
                                015F   429 ;       R0 LOW BIT SET INDICATES VERIFY SUCCESS WITH:
                                015F   430 ;
                                015F   431 ;               R1 = ADDRESS OF CCB.
                                015F   432 ;               R2 = CHANNEL INDEX.
                                015F   433 ;-
                                015F   434
                                015F   435 IOC$VERIFYCHAN::                             ;VERIFY I/O CHANNEL NUMBER
           50    0F    AA      015F   436          BICW    #CCB$C_LENGTH-1,R0          ;CLEAR EXTRANEOUS LOW ORDER BITS
                 28    13      0162   437          BEQL    10$                         ;IF EQL INVALID CHANNEL
                 50    B1      0164   438          CMPW    R0,@#CTL$GW_CHINDX          ;LEGAL CHANNEL NUMBER?
     00000000'9F               0166
                 1F    1E      0168   439          BGEQU   10$                         ;IF GEQU NO
           52    50    CE      016B   440          MNEGL   R0,R2                       ;CONVERT TO CHANNEL INDEX
     00000000'FF42    9E      0170   441          MOVAB   @CTL$GL_CCBBASE[R2],R1      ;GET ADDRESS OF CORRESPONDING CCB
                 51            0177
                 53    DC      0178   442          MOVPSL  R3                          ;READ CURRENT PSL
     53    02    16    EF      017A   443          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R3,R3 ;EXTRACT PREVIOUS MODE FIELD
                 53            017E
           50    24    3C      0181   444          MOVZWL  #SS$_NOPRIV,R0              ;ASSUME CALLER DOES NOT HAVE PRIVILEGE
        09 A1    53    91      0182   445          CMPB    R3,CCB$B_AMOD(R1)           ;CALLER HAVE PRIVILEGE TO ACCESS CHANNEL?
                 09    18      0186   446          BGEQ    20$                         ;IF GEQ NO
        05 50    00    E3      0188   447          BBCS    #0,R0,20$                   ;INDICATE SUCCESS
     50    013C  8F    3C      018C   448 10$:     MOVZWL  #SS$_IVCHAN,R0              ;SET INVALID CHANNEL
                 05    05      0191   449 20$:     RSB                                 ;
                               0192   450
                               0192   451          .END
```

```
$$T2                    = 00000005     D
ACB$B_RMOD                0000000B     D
ACB$B_TYPE                0000000A     D
ACB$C_LENGTH              00000018     D
ACB$K_LENGTH              00000018     D
ACB$L_AST                 00000010     D
ACB$L_ASTPRM              00000014     D
ACB$L_ASTQBL              00000004     D
ACB$L_ASTQFL              00000000     D
ACB$L_KAST                00000018     D
ACB$L_PID                 0000000C     D
ACB$W_SIZE                0000C008     D
CCB$B_AMOD                00000009     D
CCB$B_STS                 00000008     D
CCB$C_LENGTH              00000010     D
CCB$K_LENGTH              00000010     D
CCB$L_DIRP                0000000C     D
CCB$L_UCB                 00000000     D
CCB$L_WIND                0000U004     D
CCB$W_IOC                 0000000A     D
COLON                   = 0000003A     D
CRB$B_MASK                0000000E     D
CRB$B_TYPE                0000000A     D
CRB$C_LENGTH              00000038     D
CRB$K_LENGTH              00000038     D
CRB$L_INTD                00000014     D
CRB$L_INTD2               00000038     D
CRB$L_LINK                00000010     D
CRB$L_WQBL                00000004     D
CRB$L_WQFL                00000000     D
CRB$W_REFC                0000000C     D
CRB$W_SIZE                00000008     D
CTL$GL_CCBBASE            ********   X    02
CTL$GW_CHINDX             ********   X    02
CTL$GW_NMIOCH             ********   X    02
DDB$B_ACPCLASS            00000013     D
DDB$B_TYPE                0000000A     D
DDB$C_LENGTH              00000034     D
DDB$K_LENGTH              00000034     D
DDB$L_ACPD                00000010     D
DDB$L_DDT                 0000000C     D
DDB$L_LINK                00000000     D
DDB$L_UCB                 00000004     D
DDB$T_DRVNAME             00000024     D
DDB$T_NAME                00000014     D
DDB$W_SIZE                00000008     D
DEVCTL                    00000000 R   D    02
IOC$CVT_DEVNAM            00000009 RG  D    02
IOC$FFCHAN                0000003D RG  D    02
IOC$GL_DEVLIST            ********     X    02
IOC$SEARCHDEV             00000064 RG  D    02
IOC$SEARCHGEN             00000069 RG  D    02
IOC$UNLOCK                0000015B RG  D    02
IOC$VERIFYCHAN            0000015F RG  D    02
LOG$C_NAMLENGTH         = 00000040     D
NINE                    = 00000039     D
PCB$B_ASTACT              0000000C     D
PCB$B_ASTEN               0000000D     D
PCB$B_PRI                 0000000B     D
PCB$B_PRIB                0000002F     D
PCB$B_TYPE                0000000A     D
PCB$B_WEFC                0000002E     D
PCB$C_LENGTH              0000008C     D
PCB$K_LENGTH              0000008C     D
PCB$L_ARB                 00000084     D
PCB$L_ASTQBL              00000014     D
PCB$L_ASTQFL              00000010     D
PCB$L_EFC2P               00000058     D
PCB$L_EFC3P               0000005C     D
PCB$L_EFCS                00000050     D
PCB$L_EFCU                00000054     D
PCB$L_EFWM                0000004C     D
PCB$L_JIB                 00000078     D
PCB$L_OWNER               0000001C     D
PCB$L_PHD                 00000064     D
PCB$L_PHYPCB              00000018     D
PCB$L_PID                 00000060     D
PCB$L_PQB                 0000004C     D
PCB$L_SQBL                00000004     D
PCB$L_SQFL                00000000     D
PCB$L_STS                 00000024     D
PCB$L_UIC                 00000088     D
PCB$L_WSSWP               00000020     D
PCB$L_WTIME               00000028     D
PCB$Q_PRIV                0000007C     D
PCB$T_LNAME               00000068     D
PCB$T_TERMINAL            00000044     D
PCB$W_APTCNT              00000030     D
PCB$W_ASTCNT              00000038     D
PCB$W_BIOCNT              0000003A     D
PCB$W_BIOLM               0000003C     D
PCB$W_DIOCNT              0000003E     D
PCB$W_DIOLM               00000040     D
PCB$W_GPGCNT              00000034     D
PCB$W_GRP                 0000008A     D
PCB$W_MEM                 00000088     D
PCB$W_MTXCNT              0000000E     D
PCB$W_PPGCNT              00000036     D
PCB$W_PRCCNT              00000042     D
PCB$W_SIZE                00000008     D
PCB$W_STATE               0000002C     D
PCB$W_TMBU                00000032     D
PR$_IPL                 = 00000012     D
PSL$S_PRVMOD            = 00000002     D
PSL$V_PRVMOD            = 00000016     D
SEARCHDEV                 00000123 R   D    02
SEARCHUNIT                0000013F R   D    02
SIZ...                  = 00000002     D
SS$_IVCHAN              = 0000013C     D
SS$_IVDEVNAM            = 00000144     D
SS$_NOIOCHAN            = 000001B4     D
SS$_NONLOCAL            = 000008F0     D
SS$_NOPRIV              = 00000024     D
SS$_NORMAL              = 00000001     D
```

```
SS$_NOSUCHDEV                   =  00000908     D
SYS$FAO                            ********   X    02
UCA                             =  00000041     D
UCB$B_AMOD                         00000053     D
UCB$B_CEX                          00000077     D
UCB$B_CM1                          0000004A     D
UCB$B_CM2                          0000004B     D
UCB$B_DEVCLASS                     00000038     D
UCB$B_DEVTYPE                      00000039     D
UCB$B_DIPL                         00000052     D
UCB$B_DX_SCTCNT                    000000A6     D
UCB$B_ERTCNT                       00000070     D
UCB$B_ERTMAX                       00000071     D
UCB$B_FEX                          00000076     D
UCB$B_FIPL                         00000008     D
UCB$B_LOCSRV                       0000003C     D
UCB$B_OFFNDX                       00000094     D
UCB$B_OFFRTC                       00000095     D
UCB$B_REMSRV                       0000003D     D
UCB$B_SECTORS                      0000003C     D
UCB$B_SLAVE                        00000074     D
UCB$B_SPR                          00000075     D
UCB$B_STATE                        00000052     D
UCB$B_TRACKS                       0000003D     D
UCB$B_TT_CRFILL                    0000009D     D
UCB$B_TT_DECRF                     000000A1     D
UCB$B_TT_DELFF                     000000A2     D
UCB$B_TT_DESPEE                    000000A0     D
UCB$B_TT_DETYPE                    000000A4     D
UCB$B_TT_LFFILL                    0000009E     D
UCB$B_TT_SPEED                     0000009C     D
UCB$B_TYPE                         0000000A     D
UCB$B_VERTSZ                       0000003F     D
UCB$C_LENGTH                       00000074     D
UCB$C_MB_LENGTH                    00000090     D
UCB$C_TT_LENGTH                    000000BC     D
UCB$K_LENGTH                       00000074     D
UCB$K_MB_LENGTH                    00000090     D
UCB$K_TT_LENGTH                    000000BC     D
UCB$L_AMB                          00000054     D
UCB$L_ASTQBL                       00000010     D
UCB$L_ASTQFL                       0000000C     D
UCB$L_CPID                         0000005C     D
UCB$L_CRB                          00000020     D
UCB$L_DDB                          00000024     D
UCB$L_DEVCHAR                      00000034     D
UCB$L_DEVDEPEND                    0000003C     D
UCB$L_DPC                          00000080     D
UCB$L_DUETIM                       0000005C     D
UCB$L_DX_BFPNT                     0000009C     D
UCB$L_DX_BUF                       00000098     D
UCB$L_DX_RXDB                      000000A0     D
UCB$L_EMB                          00000078     D
UCB$L_FIRST                        00000014     D
UCB$L_FPC                          0000000C     D
UCB$L_FQBL                         00000004     D
UCB$L_FQFL                         00000000     D
UCB$L_FR3                          00000010     D
UCB$L_FR4                          00000014     D
UCB$L_IOQBL                        00000044     D
UCB$L_IOQFL                        00000040     D
UCB$L_IRP                          0000004C     D
UCB$L_LINK                         0000002C     D
UCB$L_LOGADR                       00000064     D
UCB$L_MAXBLOCK                     00000084     D
UCB$L_MB_MBX                       0000007C     D
UCB$L_MB_PORT                      0000008C     D
UCB$L_MB_RAST                      00000078     D
UCB$L_MB_SHB                       00000080     D
UCB$L_MB_WAST                      00000074     D
UCB$L_MB_WIOQBL                    00000088     D
UCB$L_MB_WIOQFL                    00000084     D
UCB$L_MEDIA                        0000008C     D
UCB$L_NT_DATSSB                    00000074     D
UCB$L_NT_INTSSB                    00000078     D
UCB$L_OPCNT                        00000060     D
UCB$L_OWNUIC                       0000001C     D
UCB$L_PID                          00000028     D
UCB$L_RQBL                         00000004     D
UCB$L_RQFL                         00000000     D
UCB$L_SVAPTE                       00000068     D
UCB$L_SVPN                         00000064     D
UCB$L_TT_DECHAR                    000000A8     D
UCB$L_TT_RDUE                      0000008C     D
UCB$L_TT_RTIMOU                    00000088     D
UCB$L_VCB                          00000030     D
UCB$T_PARTNER                      0000000C     D
UCB$W_BCNT                         0000006E     D
UCB$W_BCR                          00000096     D
UCB$W_BOFF                         0000006C     D
UCB$W_BUFQUO                       00000018     D
UCB$W_BYTESTOGO                    0000003E     D
UCB$W_CHARGE                       0000004A     D
UCB$W_CYLINDERS                    0000003E     D
UCB$W_DA                           0000008C     D
UCB$W_DC                           0000008E     D
UCB$W_DEVBUFSIZ                    0000003A     D
UCB$W_DEVSTS                       0000005A     D
UCB$W_DIRSEQ                       00000088     D
UCB$W_DSTADDR                      00000018     D
UCB$W_DX_BCR                       000000A4     D
UCB$W_ECT                          00000090     D
UCB$W_EC2                          00000092     D
UCB$W_ERRCNT                       00000072     D
UCB$W_FUNC                         0000007E     D
UCB$W_MB_SEED                      00000000     D
UCB$W_MSGCNT                       00000016     D
UCB$W_MSGMAX                       00000014     D
UCB$W_NT_CHAN                      0000007C     D
UCB$W_OFFSET                       0000008A     D
UCB$W_REFC                         00000050     D
UCB$W_SIZE                         00000008     D
UCB$W_SRCADDR                      0000001A     D
UCB$W_STS                          00000058     D
```

```
UCB$W_TT_DESIZE                   000000A5     D
UCB$W_UNIT                        00000048     D
UCB$W_VPROT                       0000001A     D
UCZ                             = 00000064     D
UNDERSCORE                      = 0000005F     D
ZERO                            = 00000030     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000192 (402.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T2	=00000005	181 (1)	181 (1)
CCB$B_AMOD	00000009		#-210 (1) #-445 (1)
CCB$C_LENGTH	00000010		#-211 (1) #-215 (1) #-436 (1)
COLON	=0000003A	66 (1)	#-289 (1)
CTL$GL_CCBBASE	00000000-XR		#-210 (1) 441 (1)
CTL$GW_CHINDX	00000000-XR		#-438 (1)
CTL$GW_NMIOCH	00000000-XR		#-212 (1)
DDB$L_LINK	00000000		322 (1) 332 (1) #-348 (1)
DDB$L_UCB	00000004		366 (1)
DDB$T_NAME	00000014		180 (1) 350 (1)
DEVCTL	00000000-R	78 (1)	175 (1)
IOC$CVT_DEVNAM	00000009-R	172 (1)	
IOC$FFCHAN	0000003D-R	209 (1)	
IOC$GL_DEVLIST	00000000-XR		322 (1) 332 (1)
IOC$SEARCHDEV	00000064-R	250 (1)	
IOC$SEARCHGEN	00000069-R	253 (1)	
IOC$UNLOCK	0000015B-R	401 (1)	
IOC$VERIFYCHAN	0000015F-R	435 (1)	
LOG$C_NAMLENGTH	=00000040		#-256 (1) #-291 (1) #-330 (1) 338 (1) 369 (1)
NINE	=00000039	67 (1)	#-294 (1)
PR$_IPL	=00000012		#-401.1 (1)
PSL$S_PRVMOD	=00000002		#-443 (1)
PSL$V_PRVMOD	=00000016		#-443 (1)
SEARCHDEV	00000123-R	347 (1)	#-323 (1) #-333 (1)
SEARCHUNIT	0000013F-R	364 (1)	#-328 (1) #-335 (1)
SS$_IVCHAN	=0000013C		#-448 (1)
SS$_IVDEVNAM	=00000144		#-314 (1)
SS$_NOIOCHAN	=000001B4		#-217 (1)
SS$_NONLOCAL	=000008F0		#-307 (1)
SS$_NOPRIV	=00000024		#-444 (1)
SS$_NORMAL	=00000001		#-219 (1)
SS$_NOSUCHDEV	=00000908		#-337 (1)
SYS$FAO	00000000-XR		181 (1)
UCA	=00000041	68 (1)	#-279 (1)
UCB$L_DDB	00000024		#-179 (1)
UCB$L_LINK	0000002C		366 (1) #-367 (1)
UCB$W_UNIT	00000048		#-181 (1) #-370 (1)
UCZ	=00000000	69 (1)	#-277 (1)
UNDERSCORE	=0000005F	70 (1)	#-295 (1)
ZERO	=00000030	71 (1)	#-292 (1) #-294 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	46 (1)	46 (1)
$CCBDEF	1	47 (1)	47 (1)
$CRBDEF	1	48 (1)	48 (1)
$DDBDEF	1	49 (1)	49 (1)
$DEFINI	1	46 (1)	46 (1) 47 (1) 48 (1) 49 (1) 50 (1)
			51 (1) 52 (1) 53 (1) 54 (1) 55 (1)
			56 (1) 57 (1) 58 (1)
$DEVDEF	1	50 (1)	50 (1)
$FAO_S	2	181 (1)	181 (1)
$IPLDEF	1	51 (1)	51 (1)
$LOGDEF	1	52 (1)	52 (1)
$PCBDEF	4	53 (1)	53 (1)
$PRDEF	4	54 (1)	54 (1)
$PRVDEF	4	55 (1)	55 (1)
$PSLDEF	2	56 (1)	56 (1)
$PUSHADR	1	181 (1)	181 (1)
$SSDEF	21	57 (1)	57 (1)
$UCBDEF	10	58 (1)	58 (1)
SETIPL	1	401.1 (1)	401.1 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.12	00:00:00.26
Command processing	141	00:00:00.79	00:00:01.62
Pass 1	781	00:00:14.58	00:00:20.65
Symbol table sort	0	00:00:01.51	00:00:01.73
Pass 2	114	00:00:02.78	00:00:05.00
Symbol table output	29	00:00:00.22	00:00:00.95
Psect synopsis output	6	00:00:00.03	00:00:00.06
Cross-reference output	23	00:00:00.23	00:00:00.29
Assembler run totals	1131	00:00:20.28	00:00:30.56

The working set limit was 1000 pages.
82323 bytes (161 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 996 non-local and 25 local symbols.
380 source lines were read in Pass 1, producing 0 object records in Pass 2.
79 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	8
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	2
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	21

1322 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) IOSPGD/UPDA=(IOSPGD.UPD,IOSPGD.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 48 APPLY ECC CORRECTION
(1) 111 CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
(1) 149 MAP VIRTUAL TO LOGICAL BLOCK
(1) 260 UPDATE TRANSFER PARAMETERS
(1) 310 SENSE DISK'S SIZE FDT ROUTINE

```
V54                     0000     .1          .TITLE  IOSRAM  *** IOSRAM random access I/O service routines
V54                     0000     .2          .IDENT  /05-02/
V54                     0000     .3          .LIST   MEB
V54                     0000     .4          .NLIST  CND
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 16-MAR-77
                        0000     27 ;
V54                     0000     .1 ; MODIFIED BY  T.L. SOUTTER  20-FEB-78
V54                     0000     .2 ;
V54                     0000     .3 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
V54                     0000     .4 ;
V54                     0000     .5 ;               Dave Butenhof   14-may-1980, Version 5.4
V54                     0000     .6 ;       02      Modified VMS V2.0 source for Supervisor environment
-3                      0000     31 ;
                        0000     32 ; NONPAGED RANDOM ACCESS MASS STORAGE I/O RELATED ROUTINES
                        0000     33 ;
                        0000     34 ; MACRO LIBRARY CALLS
                        0000     35 ;
                        0000     36
V54                     0000     .1
V54                     0000     .3
                        0000     37          $DEVDEF                         ;DEFINE DEVICE CHARACTERISTIC BITS
                        0000     38          $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                        0000     39          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000     40          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000     41          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     42          $PTEDEF                         ;DEFINE PAGE TABLE ENTRY FIELDS
                        0000     43          $RVTDEF                         ;DEFINE RVT OFFSETS
                        0000     44          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000     45          $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
                        0000     46          $WCBDEF                         ;DEFINE WCB OFFSETS
```

```
V54                             00000000    .5          .PSECT  SEP, SHR, EXE, WRT, LONG
V54                                 0000    .6
```

```
                                0000     48          .SBTTL  APPLY ECC CORRECTION
                                0000     49 ;+
                                0000     50 ; IOC$APPLYECC - APPLY ECC CORRECTION
                                0000     51 ;
                                0000     52 ; THIS ROUTINE IS CALLED TO APPLY AN ECC CORRECTION TO DATA THAT HAS BEEN
                                0000     53 ; TRANSFERED INTO MEMORY FROM A DISK DEVICE.
                                0000     54 ;
                                0000     55 ; INPUTS:
                                0000     56 ;
                                0000     57 ;       R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED UP TO, BUT NOT
                                0000     58 ;            INCLUDING, BLOCK TO BE CORRECTED (MUST BE A MULTIPLE OF 512
                                0000     59 ;            BYTES).
                                0000     60 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0000     61 ;
                                0000     62 ;       UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
                                0000     63 ;       UCB$W_EC1(R5) = STARTING BIT NUMBER OF ERROR BURST.
                                0000     64 ;       UCB$W_EC2(R5) = EXCLUSIVE OR CORRECTION PATTERN.
                                0000     65 ;       UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
                                0000     66 ;            THE TRANSFER.
                                0000     67 ;
                                0000     68 ; OUTPUTS:
                                0000     69 ;
                                0000     70 ;       THE CORRECTION PATTERN IS EXCLUSIVE OR'ED WITH THE DATA IN MEMORY
                                0000     71 ;       PROVIDING THE NECESSARY CORRECTION.
                                0000     72 ;
                                0000     73 ;       R3 IS PRESERVED ACROSS CALL.
                                0000     74 ;-
                                0000     75
-1                              0000     77 IOC$APPLYECC::                          ;APPLY ECC CORRECTION
                          18    BB 0000     78          PUSHR   #^M<R3,R4>         ;SAVE REGISTERS
                52  0090 C5    3C 0002     79          MOVZWL  UCB$W_EC1(R5),R2   ;GET STARTING BIT NUMBER OF ERROR BURST
                          52    D7 0007     80          DECL    R2                 ;CONVERT TO RELATIVE BIT NUMBER
                  52    F8 8F   8B 0009     81          BICB3   #^XF8,R2,R1        ;ISOLATE PATTERN SHIFT COUNT
                          51          000D
                    52    08    C6 000E     82          DIVL    #8,R2              ;CALCULATE RELATIVE BYTE OFFSET IN BLOCK
                    52    50    C0 0011     83          ADDL    R0,R2              ;CALCULATE RELATIVE OFFSET IN BUFFER
                50  0092 C5    3C 0014     84          MOVZWL  UCB$W_EC2(R5),R0   ;GET EXCLUSIVE OR CORRECTION PATTERN
                50  50    51    78 0019     85          ASHL    R1,R0,R0           ;SHIFT PATTERN TO PROPER POSITION
                    54    03    D0 001D     86          MOVL    #3,R4              ;SET LOOP COUNT
                  6E  A5    52    B1 0020     87 10$:     CMPW    R2,UCB$W_BCNT(R5)  ;BYTE OFFSET WITHIN RANGE?
                          47    1E 0024     88          BGEQU   40$                ;IF GEQU NO
                  51    6C  A5    3C 0026     89          MOVZWL  UCB$W_BOFF(R5),R1  ;GET BYTE OFFSET IN PAGE
                    51    52    C0 002A     90          ADDL    R2,R1              ;CALCULATE BYTE OFFSET OF TRANFER PTE
                  51    F7 8F   78 002D     91          ASHL    #-VA$S_BYTE,R1,R1  ;CALCULATE LONGWORD OFFSET TO TRANSFER PTE
                          51          0031
                53  68 B541    D0 0032     92          MOVL    @UCB$L_SVAPTE(R5)[R1],R3 ;GET TRANSFER PTE
                          03    19 0037     93          BLSS    20$                ;IF LSS VALID PTE
                      FFC4'     30 0039     94          BSBW    IOC$PTETOPFN       ;CONVERT TO VALID PTE
                  64  A5    04    C5 003C     95 20$:     MULL3   #4,UCB$L_SVPN(R5),R1 ;CALCULATE BYTE OFFSET TO SYSTEM PTE
                          51          0040
                    00    53    F0 0041     96          INSV    R3,#PTE$V_PFN,-    ;MOVE TRANSFER PTE INTO SYSTEM PAGE TABLE
                          15          0044     97                  #PTE$S_PFN,@MMG$GL_SPTBASE[R1] ;
            00000000'FF41           0045
                51    51    07    78 004B     98          ASHL    #VA$S_BYTE-2,R1,R1 ;CONVERT SVPN TO SYSTEM VIRTUAL ADDRESS
                  00  51    1F    E2 004F     99          BBSS    #VA$V_SYSTEM,R1,30$ ;SET SYSTEM VIRTUAL ADDRESS BIT
                                0053    100 30$:    INVALID R1                 ;INVALIDATE TRANSLATION BUFFER
                    3A    51    DA 0053             MTPR    R1,S^#PR$_TBIS
```

```
                52   6C A5   C1  0056   101          ADDL3   UCB$W_BOFF(R5),R2,R3    ;CALCULATE BYTE OFFSET IN BLOCK
                             53  005A
             09   00    53   F0  005B   102          INSV    R3,#VA$V_BYTE,#VA$S_BYTE,R1 ;INSERT BYTE OFFSET IN BLOCK
                             51  005F
                61    50   8C  0060   103          XORB    R0,(R1)                 ;CORRECT MEMORY BYTE
                50    F8 8F   78  0063   104          ASHL    #-8,R0,R0               ;SHIFT NEXT CORRECTION BYTE INTO PLACE
                             50  0067
                             52  D6  0068   105          INCL    R2                      ;UPDATE OFFSET IN BUFFER
                      B3 54   F5  006A   106          SOBGTR  R4,10$                  ;ANY MORE CORRECTIONS TO MAKE?
                             18  BA  006D   107 40$:     POPR    #^M<R3,R4>              ;RESTORE REGISTERS
                5A A5   01   A8  006F   108          BISW    #UCB$M_ECC,UCB$W_DEVSTS(R5) ;SET ECC CORRECTION MADE
                             05  0073   109          RSB                             ;
```

```
                                0074   111          .SBTTL  CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                                0074   112 ;+
                                0074   113 ; IOC$CVTLOGPHY - CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                                0074   114 ;
                                0074   115 ; THIS ROUTINE IS CALLED TO CONDITIONALLY CONVERT A LOGICAL BLOCK NUMBER
                                0074   116 ; TO A PHYSICAL DISK ADDRESS AND STORE THE RESULT IN THE I/O PACKET.
                                0074   117 ;
                                0074   118 ; INPUTS:
                                0074   119 ;
                                0074   120 ;       R0 = LOGICAL BLOCK NUMBER TO BE CONVERTED.
                                0074   121 ;       R3 = I/O PACKET ADDRESS.
                                0074   122 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0074   123 ;
                                0074   124 ; OUTPUTS:
                                0074   125 ;
                                0074   126 ;       IF UCB$V_NOCNVRT IS CLEAR IN UCB$W_DEVSTS, THE LOGICAL BLOCK NUMBER
                                0074   127 ;       IS CONVERTED TO A PHYSICAL DISK ADDRESS USING THE DISK GEOMETRY PARA-
                                0074   128 ;       METERS IN THE UCB. THE RESULT IS STORED IN THE MEDIA ADDRESS LONGWORD
                                0074   129 ;       OF THE I/O PACKET.
                                0074   130 ;
                                0074   131 ;       IF UCB$V_NOCNVRT IS SET, THE BLOCK NUMBER IS STORED IN THE MEDIA ADDRESS
                                0074   132 ;       LONGWORD WITHOUT CONVERSION.
                                0074   133 ;
                                0074   134 ;       R3 IS PRESERVED ACROSS CALL.
                                0074   135 ;-
                                0074   136
                                0074   137 IOC$CVTLOGPHY::                           ;CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
    34 A3   50   D0             0074   138          MOVL    R0,IRP$L_MEDIA(R3)      ;ASSUME NO CONVERSION
1D  5A A5    02   E0             0078   139          BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),10$ ;BYPASS CONVERSION IF SET
    52   3C A5   9A             007D   140          MOVZBL  UCB$L_DEVDEPEND(R5),R2  ;GET NUMBER OF SECTORS PER TRACK
              51   D4             0081   141          CLRL    R1                      ;CLEAR HIGH PART OF DIVIDEND
 50   50    52   7B             0083   142          EDIV    R2,R0,R0,IRP$L_MEDIA(R3) ;CALCULATE SECTOR NUMBER AND STORE
          34 A3                  0087
    52   3D A5   9A             0089   143          MOVZBL  UCB$L_DEVDEPEND+1(R5),R2 ;GET NUMBER OF TRACKS PER CYLINDER
 50   50    52   7B             008D   144          EDIV    R2,R0,R0,R1             ;CALCULATE TRACK AND CYLINDER
              51                  0091
    35 A3   51   90             0092   145          MOVB    R1,IRP$L_MEDIA+1(R3)    ;STORE TRACK NUMBER
    36 A3   50   B0             0096   146          MOVW    R0,IRP$L_MEDIA+2(R3)    ;STORE CYLINDER NUMBER
                 05             009A   147 10$:     RSB                             ;
```

```
                                009B   149          .SBTTL  MAP VIRTUAL TO LOGICAL BLOCK
                                009B   150 ;+
                                009B   151 ; IOC$MAPVBLK - MAP VIRTUAL TO LOGICAL BLOCK
                                009B   152 ;
                                009B   153 ; THIS ROUTINE IS CALLED TO MAP A VIRTUAL BLOCK TO A LOGICAL BLOCK USING A
                                009B   154 ; MAPPING WINDOW.
                                009B   155 ;
                                009B   156 ; INPUTS:
                                009B   157 ;
                                009B   158 ;       R0 = VIRTUAL BLOCK NUMBER.
                                009B   159 ;       R1 = NUMBER OF BYTES TO MAP.
                                009B   160 ;       R2 = ADDRESS OF WINDOW MAPPING BLOCK.
                                009B   161 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                009B   162 ;
                                009B   163 ; OUTPUTS:
                                009B   164 ;
                                009B   165 ;       R0 LOW BIT CLEAR INDICATES A TOTAL MAPPING FAILURE.
                                009B   166 ;
                                009B   167 ;               R2 = NUMBER OF UNMAPPED BYTES.
                                009B   168 ;
                                009B   169 ;       R0 LOW BIT SET INDICATES PARTIAL MAP WITH:
                                009B   170 ;
                                009B   171 ;               R1 = LOGICAL BLOCK NUMBER OF FIRST BLOCK.
                                009B   172 ;               R2 = NUMBER OF UNMAPPED BYTES.
                                009B   173 ;               R5 = UCB ADDRESS OF DEVICE UNIT (POSSIBLY MODIFIED).
                                009B   174 ;
                                009B   175 ;       R3 IS PRESERVED ACROSS CALL.
                                009B   176 ;-
                                009B   177
                                009B   178 IOC$MAPVBLK::                                ;MAP VIRTUAL TO LOGICAL BLOCK
    12    0A A2    91           009B   179          CMPB    WCB$B_TYPE(R2),#DYN$C_WCB ;SEE IF THIS IS REALLY A WINDOW
                 77    12       009F   180          BNEQ    110$                       ;IF NEQ NO
                               00A1   181 10$:     DSBINT  UCB$B_FIPL(R5)             ;SYNCHRONIZE ACCESS TO SYSTEM DATABASE
       7E    12    DB           00A1                       MFPR    S^#PR$_IPL,-(SP)
    12    0B A5    DA           00A4                       MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
                 1A    BB       00A8   182          PUSHR   #^M<R1,R3,R4>              ;SAVE REGISTERS
    53    16 A2    3C           00AA   183          MOVZWL  WCB$W_NMAP(R2),R3          ;GET COUNT OF RETRIEVAL POINTERS
                 18    13       00AE   184          BEQL    30$                        ;BRANCH IF EMPTY WINDOW
    54    20 A2    DE           00B0   185          MOVAL   WCB$L_STVBN(R2),R4         ;POINT TO STARTING VBN
       50    84    C3           00B4   186          SUBL    (R4)+,R0                   ;SUBTRACT STARTING VBN FROM DESIRED
 13 34 A5    05    E0           00B7   187          BBS     #DFV$V_SQD,UCB$L_DEVCHAR(R5),40$ ;IF SET, SEQUENTIAL DEVICE
                 0D    1F       00BC   188          BLSSU   30$                        ;BRANCH IF VBN PRECEDES WINDOW
                               00BE   189
                               00BE   190 ;
                               00BE   191 ; SCAN THE WINDOW, SUBTRACTING THE COUNT FIELD OF EACH POINTER FROM THE
                               00BE   192 ; CURRENT RELATIVE BLOCK NUMBER.
                               00BE   193 ;
                               00BE   194
       51    84    3C           00BE   195 20$:     MOVZWL  (R4)+,R1                   ;GET COUNT FIELD OF RETRIEVAL POINTER
       50    51    C2           00C1   196          SUBL    R1,R0                      ;SUBTRACT FROM RELATIVE BLOCK NUMBER
             0E    1F           00C4   197          BLSSU   50$                        ;BRANCH IF VBN LOCATED IN THIS POINTER
             84    D5           00C6   198          TSTL    (R4)+                      ;SKIP LBN FIELD OF POINTER
          F3 53    F5           00C8   199          SOBGTR  R3,20$                     ;LOOP THRU WINDOW
             50    D4           00CB   200 30$:     CLRL    R0                         ;VBN IS BEYOND WINDOW
             43    11           00CD   201          BRB     100$                       ;RETURN FAILURE
                               00CF   202
                               00CF   203 ;
```

```
                                00CF    204 ; DEVICE IS A SEQUENTIAL DEVICE. FIRST MAPPING POINTER CONTAINS THE UCB ADDRESS
                                00CF    205 ; OF THE CURRENT VOLUME THAT IS BEING PROCESSED. ALL BYTES ALWAYS MAP.
                                00CF    206 ;
                                00CF    207
                     55  64    D0  00CF    208 40$:    MOVL    (R4),R5                 ;GET CURRENT VOLUME UCB ADDRESS
                         35    11  00D2    209         BRB     80$                     ;
                                00D4    210
                                00D4    211 ;
                                00D4    212 ; FOUND THE RETRIEVAL POINTER CONTAINING THE STARTING VBN. R0 NOW
                                00D4    213 ; CONTAINS A NEGATIVE VALUE WHICH IS THE NUMBER OF BLOCKS BETWEEN
                                00D4    214 ; THE STARTING VBN AND THE END OF THE POINTER.
                                00D4    215 ;
                                00D4    216
                         50     DD  00D4    217 50$:    PUSHL   R0                      ;SAVE # BLOCKS MAPPED PAST START VBN
                     51  84     C0  00D6    218         ADDL    (R4)+,R1                ;FIRST LBN BEYOND THIS POINTER
                     50  51     C0  00D9    219         ADDL    R1,R0                   ;COMPUTE STARTING LBN
                                00DC    220
                                00DC    221 ;
                                00DC    222 ; IF THE NEXT RETRIEVAL POINTER IS CONTIGUOUS WITH THE ONE FOUND, ADD
                                00DC    223 ; IN ITS COUNT TO HANDLE THE CASE WHERE A TRANSFER SPANS TWO POINTERS.
                                00DC    224 ; NOTE THAT THE GREATEST NUMBER OF CONTIGUOUS POINTERS A TRANSFER CAN
                                00DC    225 ; SPAN IS TWO.
                                00DC    226 ;
                                00DC    227
                         53     D7  00DC    228         DECL    R3                      ;SEE IF THERE IS ANOTHER POINTER
                         0B     15  00DE    229         BLEQ    60$                     ;BRANCH IF NONE
                     53  84     3C  00E0    230         MOVZWL  (R4)+,R3                ;GET COUNT OF NEXT RETRIEVAL POINTER
                     64  51     D1  00E3    231         CMPL    R1,(R4)                 ;SEE IF THE NEXT POINTER IS CONTIGUOUS
                         03     12  00E6    232         BNEQ    60$                     ;BRANCH IF NOT
                     6E  53     C2  00E8    233         SUBL    R3,(SP)                 ;ADD TO # BLOCKS MAPPED (NEGATIVE)
                                00EB    234
                                00EB    235 ;
                                00EB    236 ; EXTRACT THE LBN AND RVN COMPONENTS OF THE STARTING ''LBN'' AND SWITCH
                                00EB    237 ; TO THE RIGHT UCB IF THIS IS A MULTI-VOLUME SET.
                                00EB    238 ;
                                00EB    239
50   18     00     EF  00EB    240 60$:    EXTZV   #0,#24,R0,R1                    ;EXTRACT LBN PART
            51         00EF
50   08     18     EF  00F0    241         EXTZV   #24,#8,R0,R0                    ;EXTRACT RVN
            50         00F4
            09     13  00F5    242         BEQL    70$                             ;BRANCH IF NOT VOLUME SET
  52   1C A2         D0  00F7    243         MOVL    WCB$L_RVT(R2),R2                ;GET RELATIVE VOLUME TABLE ADDR
55   40 A240         D0  00FB    244         MOVL    RVT$L_UCBLST-4(R2)[R0],R5       ;GET THE RIGHT UCB ADDRESS
                                0100    245
                                0100    246 ;
                                0100    247 ; SEE IF THE ENTIRE TRANSFER IS MAPPED CONTIGUOUSLY.
                                0100    248 ;
                                0100    249
6E   6E     09     78  0100    250 70$:    ASHL    #9,(SP),(SP)                    ;CONVERT TO # BYTES MAPPED
     6E     8E     C0  0104    251         ADDL    (SP)+,(SP)                      ;SUBTRACT FROM BYTES DESIRED
            02     18  0107    252         BGEQ    90$                             ;BRANCH IF NOT TOTAL MAP
            6E     D4  0109    253 80$:    CLRL    (SP)                            ;ZERO INDICATES COMPLETE MAP
 00000000'  8F     D0  0108    254 90$:    MOVL    #SS$_NORMAL,R0                  ;INDICATE SUCCESS
            50         0111
            1C     BA  0112    255 100$:   POPR    #^M<R2,R3,R4>                   ;RESTORE REGISTERS
                       0114    256         ENBINT                                  ;ALLOW INTERRUPTS
     12     8E     DA  0114                MTPR    (SP)+,S^#PR$_IPL
```

```
                        05  0117   257          RSB                                 ;
                            0118   258 110$:    BUG_CHECK STRNOTWCB,FATAL           ;STRUCTURE IS NOT A WINDOW BLOCK
  00000000'9F           16  0118                JSB     @#BUG$CHECK
4F 4E 52 54 53 00'      011E                    .ASCIC  "STRNOTWCB,FATAL"
46 2C 42 43 57 54       0124
      4C 41 54 41       012A
                        0F  011E
```

```
                                012E    260          .SBTTL  UPDATE TRANSFER PARAMETERS
                                012E    261 ;+
                                012E    262 ; IOC$UPDATRANSP - UPDATE TRANSFER PARAMETERS
                                012E    263 ;
                                012E    264 ; THIS ROUTINE IS CALLED TO UPDATE THE TRANSFER PARAMETERS AFTER A DISK ERROR
                                012E    265 ; HAS BEEN DISCOVERED BUT GOOD DATA WAS TRANSFERED.
                                012E    266 ;
                                012E    267 ; INPUTS:
                                012E    268 ;
                                012E    269 ;       R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED (MUST BE A MULTIPLE
                                012E    270 ;            OF 512 BYTES).
                                012E    271 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                012E    272 ;
                                012E    273 ;       UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
                                012E    274 ;       UCB$W_DA(R5) = CURRENT SECTOR AND TRACK ADDRESS.
                                012E    275 ;       UCB$W_DC(R5) = CURRENT CYLINDER ADDRESS.
                                012E    276 ;       UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
                                012E    277 ;            THE TRANSFER.
                                012E    278 ;
                                012E    279 ; OUTPUTS:
                                012E    280 ;
                                012E    281 ;       THE NUMBER OF BYTES REMAINING TO BE TRANSFERED, THE SYSTEM VIRTUAL
                                012E    282 ;       ADDRESS OF THE NEXT PTE, AND THE CURRENT DISK ADDRESS OF THE TRANSFER
                                012E    283 ;       ARE UPDATED.
                                012E    284 ;
                                012E    285 ;       R3 IS PRESERVED ACROSS CALL.
                                012E    286 ;-
                                012E    287
                                012E    288 IOC$UPDATRANSP::                            ;UPDATE TRANSFER PARAMETERS
 6E A5    50    A2              012E    289          SUBW    R0,UCB$W_BCNT(R5)          ;CALCULATE REMAINING BYTES TO TRANSFER
 50   F9  8F    78              0132    290          ASHL    #-7,R0,R0                  ;CALCULATE PTE LONGWORDS TO SKIP OVER
          50                    0136
 68 A5    50    C0              0137    291          ADDL    R0,UCB$L_SVAPTE(R5)        ;UPDATE SYSTEM VIRTUAL ADDRESS OF NEXT PTE
    50    04    C6              013B    292          DIVL    #4,R0                      ;CALCULATE NUMBER OF SECTORS TRANSFERED
008C C5    50    80              013E    293          ADDB    R0,UCB$W_DA(R5)            ;UPDATE SECTOR ADDRESS
                                0143    294
                                0143    295 ;
                                0143    296 ; RIPPLE CARRY FROM SECTOR TO TRACK AND FROM TRACK TO CYLINDER
                                0143    297 ;
                                0143    298
  3C A5    91                  0143    299 10$:     CMPB    UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SECTOR OVERFLOW?
008C C5                         0146
        1E    1A                0149    300          BGTRU   20$                        ;IF GTRU NO
  3C A5    82                  014B    301          SUBB    UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SUBTRACT OUT A TRACK
008C C5                         014E
008D C5    96                  0151    302          INCB    UCB$W_DA+1(R5)             ;INCREMENT TRACK ADDRESS
  3D A5    91                  0155    303          CMPB    UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;TRACK OVERFLOW?
008D C5                         0158
        E6    1A                015B    304          BGTRU   10$                        ;IF GTRU NO
  3D A5    82                  015D    305          SUBB    UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;SUBTRACT OUT A CYLINDER
008D C5                         0160
008E C5    B6                  0163    306          INCW    UCB$W_DC(R5)               ;UPDATE CYLINDER ADDRESS
        DA    11                0167    307          BRB     10$                        ;
                                0169    308 20$:     RSB                                ;
```

```
                                016A   310          .SBTTL  SENSE DISK'S SIZE FDT ROUTINE
                                016A   311 ;+
                                016A   312 ; IOC$SENSEDISK - SENSE DISK'S SIZE FDT ROUTINE
                                016A   313 ;
                                016A   314 ; THIS ROUTINE IS THE STANDARD SENSEMODE/SENSECHAR FDT ROUTINE FOR
                                016A   315 ; DISK DEVICES.  IT OBTAINS THE DISK'S SIZE, IN LOGICAL BLOCKS, FROM THE
                                016A   316 ; UCB (UCB$L_MAXBLOCK) AND IMMEDIATELY COMPLETES THE I/O REQUEST WITH
                                016A   317 ; THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE DISK'S SIZE.
                                016A   318 ;
                                016A   319 ; INPUTS:
                                016A   320 ;
                                016A   321 ;       R0 = SCRATCH.
                                016A   322 ;       R1 = SCRATCH.
                                016A   323 ;       R2 = SCRATCH.
                                016A   324 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                016A   325 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                016A   326 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                016A   327 ;       R6 = ADDRESS OF CCB.
                                016A   328 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                016A   329 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                016A   330 ;       R9 = SCRATCH.
                                016A   331 ;       R10 = SCRATCH.
                                016A   332 ;       R11 = SCRATCH.
                                016A   333 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                016A   334 ;
                                016A   335 ; OUTPUTS:
                                016A   336 ;
                                016A   337 ;       THE DISK'S SIZE, IN LOGICAL BLOCKS, IS OBTAINED FROM THE UCB
                                016A   338 ;       AND THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD
                                016A   339 ;       EQUAL TO THE DISK'S SIZE.
                                016A   340 ;-
                                016A   341
                                016A   342 IOC$SENSEDISK::                              ;SENSE DISK'S SIZE
    51    0084 C5    D0         016A   343          MOVL    UCB$L_MAXBLOCK(R5),R1      ;GET DISK'S SIZE IN LOGICAL BLOCKS
          50    00'  3C         016F   344          MOVZWL  S^#SS$_NORMAL,R0           ;SET NORMAL COMPLETION STATUS
              FE8B'  31         0172   345          BRW     EXE$FINISHIO               ;FINISH I/O OPERATION
                                0175   346
                                0175   347          .END
```

```
BUG$CHECK           ********    X   02
DEV$V_SQD         = 00000005      D
DYN$C_WCB         = 00000012      D
EXE$FINISHIO        ********    X   02
IOC$APPLYECC        00000000 RG D   02
IOC$CVTLOGPHY       00000074 RG D   02
IOC$MAPVBLK         0000009B RG D   02
IOC$PTETOPFN        ********    X   02
IOC$SENSEDISK       0000016A RG D   02
IOC$UPDATRANSP      0000012E RG D   02
IRP$B_CARCON        00000038    D
IRP$B_EFN           00000022    D
IRP$B_PRI           00000023    D
IRP$B_RMOD          0000000B    D
IRP$B_TYPE          0000000A    D
IRP$C_LENGTH        0000005C    D
IRP$K_LENGTH        0000005C    D
IRP$L_ARB           00000050    D
IRP$L_AST           00000010    D
IRP$L_ASTPRM        00000014    D
IRP$L_DIAGBUF       00000044    D
IRP$L_EXTEND        0000004C    D
IRP$L_IOQBL         00000004    D
IRP$L_IOQFL         00000000    D
IRP$L_IOSB          00000024    D
IRP$L_IOST1         00000034    D
IRP$L_IOST2         00000038    D
IRP$L_MEDIA         00000034    D
IRP$L_PID           0000000C    D
IRP$L_SEGVBN        00000040    D
IRP$L_SEQNUM        00000048    D
IRP$L_SVAPTE        0000002C    D
IRP$L_TT_TERM       00000038    D
IRP$L_UCB           0000001C    D
IRP$L_WIND          00000018    D
IRP$Q_NT_PRVMSK     0000003C    D
IRP$W_ABCNT         0000003C    D
IRP$W_BCNT          00000032    D
IRP$W_BOFF          00000030    D
IRP$W_CHAN          00000028    D
IRP$W_FUNC          00000020    D
IRP$W_OBCNT         0000003E    D
IRP$W_SIZE          00000008    D
IRP$W_STS           0000002A    D
IRP$W_TT_PRMPT      0000003C    D
MMG$GL_SPTBASE      ********    X   02
PR$_IPL           = 00000012      D
PR$_TBIS          = 0000003A      D
PTE$S_PFN         = 00000015      D
PTE$V_PFN         = 00000000      D
RVT$L_UCBLST      = 00000044      D
SIZ...            = 00000001      D
SS$_NORMAL          ********    X   02
UCB$B_AMOD          00000053    D
UCB$B_CEX           00000077    D
UCB$B_CM1           0000004A    D
UCB$B_CM2           0000004B    D
UCB$B_DEVCLASS      00000038    D
UCB$B_DEVTYPE       00000039    D
UCB$B_DIPL          00000052    D
UCB$B_DX_SCTCNT     000000A6    D
UCB$B_ERTCNT        00000070    D
UCB$B_ERTMAX        00000071    D
UCB$B_FEX           00000076    D
UCB$B_FIPL          0000000B    D
UCB$B_LOCSRV        0000003C    D
UCB$B_OFFNDX        00000094    D
UCB$B_OFFRTC        00000095    D
UCB$B_REMSRV        0000003D    D
UCB$B_SECTORS       0000003C    D
UCB$B_SLAVE         00000074    D
UCB$B_SPR           00000075    D
UCB$B_STATE         00000052    D
UCB$B_TRACKS        0000003D    D
UCB$B_TT_CRFILL     0000009D    D
UCB$B_TT_DECRF      000000A1    D
UCB$B_TT_DELFF      000000A2    D
UCB$B_TT_DESPEE     000000A0    D
UCB$B_TT_DETYPE     000000A4    D
UCB$B_TT_LFFILL     0000009E    D
UCB$B_TT_SPEED      0000009C    D
UCB$B_TYPE          0000000A    D
UCB$B_VERTSZ        0000003F    D
UCB$C_LENGTH        00000074    D
UCB$C_MB_LENGTH     00000090    D
UCB$C_TT_LENGTH     000000BC    D
UCB$K_LENGTH        00000074    D
UCB$K_MB_LENGTH     00000090    D
UCB$K_TT_LENGTH     000000BC    D
UCB$L_AMB           00000054    D
UCB$L_ASTQBL        00000010    D
UCB$L_ASTQFL        0000000C    D
UCB$L_CPID          0000005C    D
UCB$L_CRB           00000020    D
UCB$L_DDB           00000024    D
UCB$L_DEVCHAR       00000034    D
UCB$L_DEVDEPEND     0000003C    D
UCB$L_DPC           00000080    D
UCB$L_DUETIM        0000005C    D
UCB$L_DX_BFPNT      0000009C    D
UCB$L_DX_BUF        00000098    D
UCB$L_DX_RXDB       000000A0    D
UCB$L_EMB           00000078    D
UCB$L_FIRST         00000014    D
UCB$L_FPC           0000000C    D
UCB$L_FQBL          00000004    D
UCB$L_FQFL          00000000    D
UCB$L_FR3           00000010    D
UCB$L_FR4           00000014    D
UCB$L_IOQBL         00000044    D
UCB$L_IOQFL         00000040    D
UCB$L_IRP           0000004C    D
UCB$L_LINK          0000002C    D
UCB$L_LOGADR        00000064    D
UCB$L_MAXBLOCK      00000084    D
UCB$L_MB_MBX        0000007C    D
UCB$L_MB_PORT       0000008C    D
UCB$L_MB_RAST       00000078    D
UCB$L_MB_SHB        00000080    D
UCB$L_MB_WAST       00000074    D
UCB$L_MB_WIOQBL     00000088    D
UCB$L_MB_WIOQFL     00000084    D
UCB$L_MEDIA         0000008C    D
UCB$L_NT_DATSSB     00000074    D
UCB$L_NT_INTSSB     00000078    D
UCB$L_OPCNT         00000060    D
UCB$L_OWNUIC        0000001C    D
UCB$L_PID           00000028    D
UCB$L_RQBL          00000004    D
UCB$L_RQFL          00000000    D
UCB$L_SVAPTE        00000068    D
UCB$L_SVPN          00000064    D
UCB$L_TT_DECHAR     000000A8    D
UCB$L_TT_RDUE       0000008C    D
UCB$L_TT_RTIMOU     000000B8    D
UCB$L_VCB           00000030    D
UCB$M_ECC         = 00000001    D
UCB$T_PARTNER       0000000C    D
UCB$V_NOCNVRT     = 00000002    D
UCB$W_BCNT          0000006E    D
UCB$W_BCR           00000096    D
UCB$W_BOFF          0000006C    D
UCB$W_BUFQUO        00000018    D
UCB$W_BYTESTOGO     0000003E    D
UCB$W_CHARGE        0000004A    D
UCB$W_CYLINDERS     0000003E    D
UCB$W_DA            0000008C    D
UCB$W_DC            0000008E    D
UCB$W_DEVBUFSIZ     0000003A    D
UCB$W_DEVSTS        0000005A    D
UCB$W_DIRSEQ        00000088    D
UCB$W_DSTADDR       00000018    D
UCB$W_DX_BCR        000000A4    D
UCB$W_ECT           00000090    D
UCB$W_EC2           00000092    D
UCB$W_ERRCNT        00000072    D
UCB$W_FUNC          0000007E    D
UCB$W_MB_SEED       00000000    D
UCB$W_MSGCNT        00000016    D
UCB$W_MSGMAX        00000014    D
UCB$W_NT_CHAN       0000007C    D
UCB$W_OFFSET        0000008A    D
UCB$W_REFC          00000050    D
UCB$W_SIZE          00000008    D
UCB$W_SRCADDR       0000001A    D
UCB$W_STS           00000058    D
UCB$W_TT_DESIZE     000000A5    D
UCB$W_UNIT          00000048    D
UCB$W_VPROT         0000001A    D
VA$S_BYTE         = 00000009    D
VA$V_BYTE         = 00000000    D
```

```
VA$V_SYSTEM     = 0000001F     D
WCB$B_ACCESS      0000000B     D
WCB$B_TYPE        0000000A     D
WCB$C_LENGTH      00000024     D
WCB$C_MAP         00000024     D
WCB$K_LENGTH      00000024     D
WCB$K_MAP         00000024     D
WCB$L_FCB         00000018     D
WCB$L_LBN         00000002     D
WCB$L_ORGUCB      00000010     D
WCB$L_P1_LBN      00000026     D
WCB$L_P2_LBN      0000002C     D
WCB$L_PID         0000000C     D
WCB$L_PREVLBN     FFFFFFFC     D
WCB$L_RVT         0000001C     D
WCB$L_STVBN       00000020     D
WCB$L_WLBL        00000004     D
WCB$L_WLFL        00000000     D
WCB$W_ACON        00000014     D
WCB$W_COUNT       00000000     D
WCB$W_NMAP        00000016     D
WCB$W_P1_COUNT    00000024     D
WCB$W_P2_COUNT    0000002A     D
WCB$W_PREVCOUNT   FFFFFFFA     D
WCB$W_REFCNT      0000000E     D
WCB$W_SIZE        00000008     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000175 (373.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BUG$CHECK	00000000-XR		258 (1)
DEV$V_SQD	=00000005		#-187 (1)
DYN$C_WCB	=00000012		#-179 (1)
EXE$FINISHIO	00000000-XR		#-345 (1)
IOC$APPLYECC	00000000-R	77 (1)	
IOC$CVTLOGPHY	00000074-R	137 (1)	
IOC$MAPVBLK	0000009B-R	178 (1)	
IOC$PTETOPFN	00000000-XR		#-94 (1)
IOC$SENSEDISK	0000016A-R	342 (1)	
IOC$UPDATRANSP	0000012E-R	288 (1)	
IRP$L_MEDIA	00000034		#-138 (1) #-142 (1) #-145 (1) #-146 (1)
MMG$GL_SPTBASE	00000000-XR		97 (1)
PR$_IPL	=00000012		#-181 (1) #-256 (1)
PR$_TBIS	=0000003A		#-100 (1)
PTE$S_PFN	=00000015		#-97 (1)
PTE$V_PFN	=00000000		#-96 (1)
RVT$L_UCBLST	=00000044		#-244 (1)
SS$_NORMAL	00000000-XR		#-254 (1) #-344 (1)
UCB$B_FIPL	0000000B		#-181 (1)
UCB$L_DEVCHAR	00000034		187 (1)
UCB$L_DEVDEPEND	0000003C		#-140 (1) #-143 (1) #-299 (1) #-301 (1) #-303 (1) #-305 (1)
UCB$L_MAXBLOCK	00000084		#-343 (1)
UCB$L_SVAPTE	00000068		#-291 (1) #-92 (1)
UCB$L_SVPN	00000064		#-95 (1)
UCB$M_ECC	=00000001		#-108 (1)
UCB$V_NOCNVRT	=00000002		#-139 (1)
UCB$W_BCNT	0000006E		#-289 (1) #-87 (1)
UCB$W_BOFF	0000006C		#-101 (1) #-89 (1)
UCB$W_DA	0000008C		#-293 (1) #-299 (1) #-301 (1) #-302 (1) #-303 (1) #-305 (1)
UCB$W_DC	0000008E		#-306 (1)
UCB$W_DEVSTS	0000005A		#-108 (1) 139 (1)
UCB$W_EC1	00000090		#-79 (1)
UCB$W_EC2	00000092		#-84 (1)
VA$S_BYTE	=00000009		#-102 (1) #-91 (1) #-98 (1)
VA$V_BYTE	=00000000		#-102 (1)
VA$V_SYSTEM	=0000001F		#-99 (1)
WCB$B_TYPE	0000000A		#-179 (1)
WCB$L_RVT	0000001C		#-243 (1)
WCB$L_STVBN	00000020		185 (1)
WCB$W_NMAP	00000016		#-183 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	37 (1)	37 (1) 38 (1) 39 (1) 40 (1) 41 (1) 42 (1) 43 (1) 44 (1) 45 (1) 46 (1)
$DEVDEF	1	37 (1)	37 (1)
$DYNDEF	2	38 (1)	38 (1)
$IPLDEF	1	39 (1)	39 (1)
$IRPDEF	4	40 (1)	40 (1)
$PRDEF	4	41 (1)	41 (1)
$PTEDEF	3	42 (1)	42 (1)
$RVTDEF	1	43 (1)	43 (1)
$UCBDEF	10	44 (1)	44 (1)
$VADEF	1	45 (1)	45 (1)
$WCBDEF	3	46 (1)	46 (1)
BUG_CHECK	1	258 (1)	258 (1)
DSBINT	1	181 (1)	181 (1)
ENBINT	1	256 (1)	256 (1)
INVALID	1	100 (1)	100 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.13	00:00:00.49
Command processing	134	00:00:00.75	00:00:02.01
Pass 1	700	00:00:11.81	00:00:17.44
Symbol table sort	0	00:00:00.72	00:00:01.63
Pass 2	92	00:00:02.12	00:00:02.90
Symbol table output	17	00:00:00.17	00:00:00.44
Psect synopsis output	6	00:00:00.02	00:00:00.02
Cross-reference output	22	00:00:00.22	00:00:00.41
Assembler run totals	1008	00:00:15.95	00:00:25.36

The working set limit was 1000 pages.
65201 bytes (128 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 583 non-local and 18 local symbols.
362 source lines were read in Pass 1, producing 0 object records in Pass 2.
88 pages of virtual memory were used to define 23 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	6
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	6
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	19

877 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) IOSRAM/UPDA=(IOSRAM.UPD,IOSRAM.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 450 Libraries and Macros
(1) 496 Equated Symbols
(1) 561 Data Psect Declarations
(1) 580 Work Psect Declarations
(1) 712 Data Psect Declarations
(5) 733 SHELL OF STAND ALONE SUPERVISOR MEMORY.
(5) 805 DIAGNOSTIC SUPERVISOR BOOTSTRAP ROUTINE.
(5) 1010 USEP KERNEL ROUTINE
(5) 1311 COMMON KERNEL REFRESH ENTRY POINT.
(6) 1479 INIT_CONTEXT Initialize process context
(6) 1605 CONTROL-C AST HANDLER
(6) 1652 KEYBOARD CHECK ROUTINE
(6) 1806 DSX$CNTRLC Program enable for Control-C intercept
(6) 1855 DSV$EXIT, Process EXIT command
(6) 1910 EXIT$HANDLR Process EXIT handler
(6) 1976 DSR$Check_MenuTest_Off_Set Routine
(6) 2056 DSR$Check_AutoTest_Off_Set Routine
(8) 2130 DSX$GETTERM Get User Terminal Characteristics [87]

```
0000      1          .Title  KERNEL  *** KERNEL Main control routine
0000      2          .Ident  /07-97/
0000      3          .NoShow Conditionals
0000      4
0000      5 ;++
0000      6 ; Copyright (c) 1977, 1983, 1984
0000      7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      8 ;
0000      9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000     14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     15 ; REMAIN IN DEC.
0000     16 ;
0000     17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     19 ; CORPORATION.
0000     20 ;
0000     21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     23 ;--
```

```
                        0000     25 ;++
                        0000     26 ; Facility:
                        0000     27 ;
                        0000     28 ;       VAX Diagnostic Supervisor.
                        0000     29 ;
                        0000     30 ; Abstract:
                        0000     31 ;
                        0000     32 ;       This module contains the main control routines for the VAX
                        0000     33 ;       Diagnostic Supervisor.
                        0000     34 ;
                        0000     35 ; Environment:
                        0000     36 ;
                        0000     37 ;       Not applicable.
                        0000     38 ;
                        0000     39 ; Author:
                        0000     40 ;
                        0000     41 ;       Ken Chapman
                        0000     42 ;
                        0000     43 ; Date:
                        0000     44 ;
                        0000     45 ;       26-Oct-77
                        0000     46 ;
                        0000     47 ; Version:
                        0000     48 ;
                        0000     49 ;       Version 01.
                        0000     50 ;
```

```
                                0000     52 ;
                                0000     53 ; Modified By:
                                0000     54 ;       01      - Nick Howgate, 15-nov-1977, Version 02.
                                0000     55 ;                 APT INTERFACE UPDATES.
                                0000     56 ;
                                0000     57 ;       02      - Tom Soutter, 21-nov-1977, Version 03.
                                0000     58 ;                 DSPR #14  DEFAULT FLAG SETTING.
                                0000     59 ;
                                0000     60 ;       03      - Ken Chapman, 06-dec-1977, Version 04.
                                0000     61 ;                 REMOVED MEMORY MAPS.
                                0000     62 ;
                                0000     63 ;       04      - Ken Chapman, 06-dec-1977, Version 04.
                                0000     64 ;                 DSPR #3  MODIFIED ^C BREAK TO CLI.
                                0000     65 ;
                                0000     66 ;       05      - Nick Howgate, 19-dec-1977, Version 05.
                                0000     67 ;                 MADE AX_BUFF, USTKPTR, SSTKPTR, ESTKPTR GLOBAL FOR MEMMGT.
                                0000     68 ;
                                0000     69 ;       06      - Nick Howgate, 19-dec-1977, Version 05.
                                0000     70 ;                 MADE ALL STACKS 2 PAGES.
                                0000     71 ;
                                0000     72 ;       07      - Nick Howgate, 30-Dec-1977, Version 06 (ESSAA-3.06).
                                0000     73 ;                 DELETE SYMBOL DS$GL_PASSES
                                0000     74 ;
                                0000     75 ;       08      - Nick Howgate, 06-Feb-1978, Version 07 (ESSAA-4.00).
                                0000     76 ;                 INCLUDE QIO SUPPORT
                                0000     77 ;
                                0000     78 ;                 - Tom Soutter, 03-Mar-1978
                                0000     79 ;       09        ADDED SOFTWARE PCB FOR AST HANDLING
                                0000     80 ;       10        FIXED SOME BUGS IN PCB SETUP CODE
                                0000     81 ;       11        DEFINED DIAGNOSTIC PID, SET UP SAME IN SOFTWARE PCB
                                0000     82 ;
                                0000     83 ;       12      - Ken Chapman, 23-Mar-1978, Version 08 (ESSAA-4.01).
                                0000     84 ;                 ADDED BUFFER INIT CODE.
                                0000     85 ;                 REMOVED MOVC5 AND LDPCTX INSTRUCTIONS.
                                0000     86 ;
                                0000     87 ;                 - Tom Soutter, 30-Mar-1978
                                0000     88 ;       13        ADDED CODE TO INITIALIZE THE SYSTEM TIMER
                                0000     89 ;       14        REMOVED STACK BUFFER PAGES, REDUCED USER STACK TO ONE
                                0000     90 ;                 PAGE AND OVER-MAPPED THE SUPERVISOR & EXECUTIVE STACKS
                                0000     91 ;                 ONTO THE USER STACK
                                0000     92 ;
                                0000     93 ;       15      - Nick Howgate, 18-May-1978, Version 09 (ESSAA-4.02).
                                0000     94 ;                 PLACE IPL LEVEL AT 1F AT SYSTEM START UP
                                0000     95 ;                 LOWER IPL LEVEL ONLY AFTER SOFTWARE INITIALIZATION
                                0000     96 ;
                                0000     97 ;       16      - Nick Howgate, 09-Jun-1978, Version 10 (ESSAA-4.03).
                                0000     98 ;                 FIX APT ^C PROBLEM
                                0000     99 ;
                                0000    100 ;       17      - Nick Howgate, 14-Jun-1978
                                0000    101 ;                 CLEAR EVENTS FLAGS AT SUPERVISOR START
```

```
0000   103 ;                - Roger Riggs, 10-Aug-1978
0000   104 ;          18    Revised basic program loop
0000   105 ;          19    Fix for changed position of year in DS$GT_NEWYEAR
0000   106 ;                Added separate allocations for USER, EXEC, and SUPER stacks
0000   107 ;          20    Added call to INI$PTABLE to initialize P-table list at
0000   108 ;                supervisor start.
0000   109 ;          21    Added code to Auto configure/ATTACH the default load device
0000   110 ;                when loaded with DIAGBOOT.
0000   111 ;          22    Added process exit handler to execute use cleanup and
0000   112 ;                deallocate device allocated by select.
0000   113 ;
0000   114 ;          23    - Roger Riggs, 31-Aug-1979
0000   115 ;                Added initial VDT breakpoint and saved VMB boot flags
0000   116 ;
0000   117 ;          24    - Roger Riggs, 21-Jan-1980, Version 5.2
0000   118 ;                Added code to always prompt APT on start, also copy
0000   119 ;                processor type code to EXE$GB_CPUTYPE
0000   120 ;
0000   121 ;          25    - David R. Butenhof, 06-feb-1980
0000   122 ;                Routine INIT_CONTEXT should force AST delivery enable.
0000   123 ;
0000   124 ;          26    - John Ciukaj, 25-Jan-1980, Version 5.2
00C0   125 ;                Added location BOOT$L_DEVTYP to be loaded from RPB$B_DEVTYP.
0000   126 ;
0000   127 ;          27    - Roger Riggs, 12-Mar-1980
0000   128 ;                Clear DS$GB_MM_ENB on START 10000, as well as memory managment
0000   129 ;
0000   130 ;                - Roger Riggs, 24-Mar-1980
0000   131 ;          28    Added code to verify that system identification register
0000   132 ;                agrees with current supervisor.
0000   133 ;          29    Added code to type message and exit if SYS$INPUT is not
0000   134 ;                a terminal or a mailbox.  Fix for SPR#11-28511.
0000   135 ;                Later fix will use RMS for SYS$INPUT I/O in usermode.
0000   136 ;
0000   137 ;                - Dave Butenhof, -Mar-1980
0000   138 ;          30    Add RMS setup in usermode, to allow supervisor execution
0000   139 ;                via VMS command file/batch.
0000   140 ;          31    Fix CTL$GL_CCBBASE label define to point to a pointer
0000   141 ;                longword.  And fix order of arguments to PRINT on SID
0000   142 ;                mis-match.
0000   143 ;
0000   144 ;          32    - Roger Riggs, 21-May-1980, Version 5.4
0000   145 ;                Changed order of stacks to allow for memory protection
0000   146 ;                when MM is on
```

```
                        0000   148 ;
                        0000   149 ;       33      Moved IOC$GL_PSFL/BL to Entry
                        0000   150 ;       34      Added JIB
                        0000   151 ;
                        0000   152 ;       35      - Dave Butenhof, 28-aug-1980, Version 6.0
                        0000   153 ;               Add call to RMS$INIT in standalone initialization to
                        0000   154 ;               clear RMS file table.
                        0000   155 ;
                        0000   156 ;               - Roger Riggs, 08-Sept-1980, Version 6.0
                        0000   157 ;       36      Added code to set and restore on exit the default directory
                        0000   158 ;               and disk.
                        0000   159 ;       37      Move translation of load device name to DEF$Q_DEV on
                        0000   160 ;               usemode entry
```

```
                        0000   162 ;
                        0000   163 ;       38      - Roger Riggs, 17-Oct-1980
                        0000   164 ;               Made reference to EXE$GQ_SYSTIME L^ offset.
                        0000   165 ;
                        0000   166 ;               - Dave Butenhof, 17-oct-1980, version 6.1
                        0000   167 ;       39      Add new argument to DSX$CNTRLC; DISABL--if low bit is set,
                        0000   168 ;               disables control C handling, if clear, enables it.  Also
                        0000   169 ;               KB_CHECK ignores control C flag if handling is disabled.
                        0000   170 ;               Also, if user control C routine return failure code, the
                        0000   171 ;               Supervisor will proceed to handle the control C normally
                        0000   172 ;               itself.
                        0000   173 ;       40      Add label BEGIN_BLISS (same as BEGIN) for BLISS routines to
                        0000   174 ;               use after INIT_CONTEXT, since BEGIN is reserved word.
```

```
                                0000    176 ;
                                0000    177 ;       41      - Dave Butenhof, 18-feb-1981, version 6.3
                                0000    178 ;               Delete use of SEP psect.
                                0000    179 ;
                                0000    180 ;       42      - Dave Butenhof, 16-mar-1981, version 6.3
                                0000    181 ;               Fix ^C handling; move flag clears to make user status=0
                                0000    182 ;               return work under APT.
                                0000    183 ;
                                0000    184 ;       43      - Dave Butenhof, 23-mar-1981, version 6.3
                                0000    185 ;               Expand Exec/Supervisor stack to 3 pages each.
```

```
0000   187 ;        44      - Jack Stansbury, 21-Oct-1981, Version 6.5
0000   188 ;                Added calls to QA$MAIN for the QA enhancement.
0000   189 ;                Also added .LIBRARY statements for $DS and $DIAG.
0000   190 ;                Also changed one of the error messages to lower case.
0000   191 ;                Also fixed some truncation errors.
0000   192 ;
0000   193 ;        45      - Jack Stansbury, 10-Nov-1981, Version 6.5
0000   194 ;                Changed ALL W^ to L^.
0000   195 ;
0000   196 ;        46      - Dave Butenhof, 12-Nov-1981, version 6.5
0000   197 ;                Add DS$GB_TYPECODE for 'SET BINARY' function.
0000   198 ;
0000   199 ;        47      - Dave Butenhof, 13-Nov-1981, version 6.5
0000   200 ;                Implement EXIT command (DSV$EXIT).  It will do an
C000   201 ;                $EXIT in user mode, or a HALT in standalone; after
0000   202 ;                doing program cleanup, etc.
0000   203 ;
0000   204 ;        48      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000   205 ;                Commented out several calls to QA_MAIN because it appears as
0000   206 ;                though they will not be needed for the 6.5 DS version.
```

```
                        0000   208 ;
                        0000   209 ;       49      - Dave Butenhof, 14-Dec-1981, version 6.6
                        0000   210 ;               Modify startup to use $GETSYI call in usermode, to get
                        0000   211 ;               the actual processor SID into DSA$GL_SID.
                        0000   212 ;
                        0000   213 ;       50      - Dave Butenhof, 29-Dec-1981, version 6.6
                        0000   214 ;               Add locations to store error counts for HARD, SOFT, DEVICE,
                        0000   215 ;               SYSTEM, and SOFTWARE (ERRSUP) errors.
                        0000   216 ;
                        0000   217 ;       51      - Jack Stansbury, 10-Feb-1982, Version 6.6X
                        0000   218 ;               Added another Control-C handler for the DS to use. This is for
                        0000   219 ;               QA to trap control-c's, but can be used by any other
                        0000   220 ;               DS-integrated facility. See the KB_Check routine for the code.
                        0000   221 ;
                        C000   222 ;       52      - Dave Butenhof, 09-Mar-1982, version 6.6
                        0000   223 ;               Add support for rooted systems.  When booted from DIAGBOOT,
                        0000   224 ;               startup code will do 'SET LOAD' command on the filespec in the
                        0000   225 ;               RPB$T_FILE field.
```

```
                     0000   227 ;       53      - Dave Butenhof, 26-Mar-1982, version 6.7
                     0000   228 ;               Fix rooted system directory support...redefine SYS$DISK when
                     0000   229 ;               translating SYS$SYSROOT.  SYS$SYSROOT:[SYSO,SYSMAINT] don't
                     0000   230 ;               Added SYS$LIBRARY:LIB.
                     0000   231 ;
                     0000   232 ;       54      Marion Baggett, 31-Mar-1982, Version 6.7
                     0000   233 ;               Added SYS$LIBRARY:LIB
                     0000   234 ;
                     0000   235 ;       55      Marion Baggett, 2-Apr-1982, Version 6.7
                     0000   236 ;               Added DS$GL_PREPERR_COUNT field to count device preparation errors.
                     0000   237 ;
                     0000   238 ;       56      Marion Baggett, 5-Apr-1982, Version 6.7
                     0000   239 ;               Changed $DS_PRINT to $PRINT macros.
                     000C   240 ;
                     C000   241 ;       57      Marion Baggett, 7-Apr-1982, Version 6.7
                     0000   242 ;               Added $DS_TYPEDEF to define type codes.
                     0000   243 ;
                     0000   244 ;       58      Dave Butenhof, 08-Apr-1982, version 6.7
                     0000   245 ;               Fix truncation error.
                     0000   246 ;
                     0000   247 ;       59      Jack Stansbury, 9-Apr-1982, Version 6.7
                     0000   248 ;               Added $CRD Library statement. Added support for the
                     0000   249 ;               Customer Runnable Diagnostic package.
                     0000   250 ;
                     0000   251 ;       60      Dave Butenhof, 13-Apr-1982, version 6.7
                     0000   252 ;               Add setup of DS$GB_INHIBIT_NAMING, to enable/disable
                     0000   253 ;               generic name enforcement. (Set in APT mode).
```

```
                        0000   255 ;
                        0000   256 ;       61      Dave Butenhof, 17-May-1982, version 6.8
                        0000   257 ;               Fix size of Kernel stack (make it larger to minimize trouble).
                        0000   258 ;
                        0000   259 ;       62      Dave Butenhof, 21-May-1982, version 6.8
                        0000   260 ;               Automatically attach console device at startup, to make
                        0000   261 ;               RMS and DIRECTORY a bit simpler (no special case for device
                        0000   262 ;               without Ptable).
                        0000   263 ;
                        0000   264 ;       63      Jack Stansbury, 7-June-1982, Version 6.8
                        0000   265 ;               Added code in DSV$Exit routine that will not allow the user
                        0000   266 ;               to Continue (console command) once a HALT instruction has been
                        0000   267 ;               executed - only if CRD-AutoTest bits are set in the R5 passed
                        0000   268 ;               to the DS.
                        C000   269 ;
                        0000   270 ;       64      Jack Stansbury, 9-Jun-1982, Version 6.8
                        0000   271 ;               Added code in the DSV$Exit routine that will allow output of
                        0000   272 ;               pertinent CRD variables when a CRD-Internal error occurs. This
                        0000   273 ;               is indicated by CRD changing the byte in its own dispatch vector
                        0000   274 ;               area (DS$GA_CRD_Starting_Address + 1) to be one more than the absolu
                        0000   275 ;               value of its dispatch vectors + 2 byte. That is, CRD will increment
                        0000   276 ;               the "positive version number" in its dispatch vector area by one whe
                        0000   277 ;               it detects a internal error. CRD will then put the values of a
                        0000   278 ;               select number of pertinent variables into its own dispatch vector
                        0000   279 ;               area. DSV$Exit will print these out when the user types a console
                        0000   280 ;               Continue command.
                        0000   281 ;
                        0000   282 ;       65      Marion Baggett, 9-June-1982, Version 6.8
                        0000   283 ;               Increase the size of the interupt stack to 4 pages so the
                        0000   284 ;               Vax Architecture exerciser will run.
                        0000   285 ;
                        0000   286 ;       66      Richard Brown, 15-June-82, Version 6.8
                        0000   287 ;               Addition of IMAGE_HEADER, which is a buffer to hold the
                        0000   288 ;               image header of a diagnostic file, if the file was linked
                        0000   289 ;               with a header.  The header is used by the debugger.
                        0000   290 ;
                        0000   291 ;       67      Jack Stansbury, 28-June-1982, Version 6.8
                        0000   292 ;               Disabled ^C's in stand-alone initialization code so that CRD will
                        0000   293 ;               have a chance to set up a ^C handler. This takes care of the
                        0000   294 ;               problem of ^C's being typed while DIAGBOOT is running, and then
                        0000   295 ;               having the DS see the ^C before CRD is loaded in (thus causing the
                        0000   296 ;               DS> prompt to appear without the DS_Start message being printed).
                        0000   297 ;               ^C's are re-enabled in the Begin_0 code.
```

```
0000   299 ;
0000   300 ;       68      Richard Brown, 16-July-82, Version 6.9
0000   301 ;               Addition of code in INIT_CONTEXT to see if the T-bit is set
0000   302 ;               in the PSL and if so to set it in the new PSL.  This is so
0000   303 ;               that if the debugger has set the T-bit, it will not be cleared
0000   304 ;               out when the context is initialized.
0000   305 ;
0000   306 ;       69      Jack Stansbury, 24-July-1982, Version 6.9
0000   307 ;               Added two new routine that return a 1 in R0 iff the correct bits are
0000   308 ;               set in Boot$L_R5 for CRD_AutoTest and for CRD_MenuTest. Used these
0000   309 ;               routines in the Begin0 part.
0000   310 ;
0000   311 ;       70      Jack Stansbury, 27-July-1982, Version 6.9
0000   312 ;               Added another ^C handler for the VDS. This allows CRD Menu Test to
0000   313 ;               let the User handle a ^C if the user has established a handler
0000   314 ;               routine; but at the same time, if this user routine returns failure
0000   315 ;               meaning they did not want to handle it, let the "third" handler (VDS
0000   316 ;               second handler) have it. Leave DS$Cli as the last chance handler.
0000   317 ;
0000   318 ;       71      Jack Stansbury, 8-September-1982, Version 6.9
0000   319 ;               Made some changes to the DSV$Exit routine for CRD Menu Test. Also
0000   320 ;               made a few changes to the routines mentioned in 69 above - they were
0000   321 ;               looking at the wrong bits.
0000   322 ;
0000   323 ;       72      Jack Stansbury, 15-September-1982, Version 6.9
0000   324 ;               Cleared out the three ^C handler addresses in the VDS stand-alone
0000   325 ;               initialization code.
0000   326 ;
0000   327 ;       73      Jack Stansbury, September 15, 1982, Version 6.9
0000   328 ;               Made the DS$L_UserCntrlC longword global, for use by the CLI routine
0000   329 ;
0000   330 ;       74      Jack Stansbury, September 23, 1982, Version 6.9
0000   331 ;               Added a print statement to be printed if the CRD initialization fail
0000   332 ;
0000   333 ;       75      Jack Stansbury, September 24, 1982, Version 6.9
0000   334 ;               Added code to check to see if both CRD bits were set in the R5 passe
0000   335 ;               to the VDS from Diagboot. If so, print an error message and exit.
0000   336 ;
0000   337 ;       76      Jack Stansbury, September 28, 1982, Version 6.9
0000   338 ;               Added code to take care of the following: Start up the VDS. Type CRD
0000   339 ;               Hit ^P while CRD is running - get back to ">>>" console prompt. Type
0000   340 ;               S 10000 - get a DS> prompt. Type CRD again, but the CRD image is
0000   341 ;               already loaded. This was causing a halt at PC = 00000001. So, the
0000   342 ;               solution is to call the Unload_CRD routine when the bits are set in
0000   343 ;               the DSA flags longword when the Supervisor is being inited (it is
0000   344 ;               normally inited without any of the CRD bits being set).
0000   345 ;
0000   346 ;       77      Bob Bergazzi    7-Oct-82        Version 6.9
0000   347 ;               Changed the way system service vectors are handled; after we copy
0000   348 ;               the VMS system service vectors to 10200 (in order to get the
0000   349 ;               correct word mask), modify each entry point to immediately JMP
0000   350 ;               to the corresponding entry point in the real system service vector
0000   351 ;               table at 80000xxx. This solves the problem of code in the system
0000   352 ;               service vectors which branches around outside of the three pages
0000   353 ;               which we were copying before.
0000   354 ;
0000   355 ;       78      Bob Bergazzi    24-Nov-82       Version 6.10
```

```
                                0000    356 ;               Added a call to load the PCS on startup (DS$_LOADPCS_S)
                                0000    357 ;
                                0000    358 ;       79      Jack Stansbury, 4-Mar-83, Version 6.11
                                0000    359 ;               Changed the code in the DSR$Check routines (for CRD) - they must
                                0000    360 ;               check the DSA bits while online, not the Boot$L_R5 longword.
                                0000    361 ;
                                0000    362 ;       80      John Ciukaj,    2-Apr.-1983     Version 6.11
                                0000    363 ;               - Changed code that referenced CRD bits in DSA Flags.
                                0000    364 ;                 The bits were redefined to distinguish between
                                0000    365 ;                 online and offline.
                                0000    366 ;               - Changed name of DSR$Check_... routines to distinguish
                                0000    367 ;                 between Online and Offline.
                                0000    368 ;
                                0000    369 ;       81      Bob Bergazzi    8-Apr-1983      Version 6.11
                                0000    370 ;               Made on-line mode start up with terminal width set to 80.
                                0000    371 ;
                                0000    372 ;       82      Bob Bergazzi    25-April-1983   Version 6.11
                                0000    373 ;               During boot of ESSAA, we must determine if this is 780 or a
                                0000    374 ;               785, and if it is a 785, then we need to change
                                0000    375 ;               the clock overhead factor (DS$GK_USOVR) in the WAITUS routine.
                                0000    376 ;               This is done by referencing the symbol DSX$WAITUS_USOVR which
                                0000    377 ;               addresses the beginning of the instruction which uses the
                                0000    378 ;               overhead factor, and modifying the instruction. This is done
                                0000    379 ;               in order to avoid adding more instructions to the DSX$WAITUS
                                0000    380 ;               routine which would change the value of the DS$GK_USOVR for
                                0000    381 ;               existing processors.
                                0000    382 ;
                                0000    383 ;       83      Bob Bergazzi    25-April-1983   Version 6.11
                                0000    384 ;               Increased SCB size to 4 pages for XXX. Related changes in
                                0000    385 ;               SCB module.
                                0000    386 ;
                                0000    387 ;       84      John Ciukaj     29-Apr-83       version 6.11
                                0000    388 ;                               13-May-83
                                0000    389 ;               DSR$Check.. routines: check both DSA Flags and RPB
                                0000    390 ;               regardless of Standalone or Online.
                                0000    391 ;
                                0000    392 ;       85      Bob Bergazzi    May 16,1983     Version 6.11
                                0000    393 ;               Changed the order of the .LIB statements.
                                0000    394 ;
                                0000    395 ;       86      M.Baggett       May 16, 1983    Version 6.11
                                0000    396 ;               Get RPB$L_BOOTR2 field from RPB.
                                0000    397 ;
                                0000    398 ;       87      Richard Brown   June 22, 1983   Version 6.12
                                0000    399 ;               Added code to fetch user terminal characteristics
                                0000    400 ;               and store them in $DSGL_TERMCHAR so the new service
                                0000    401 ;               $DS_GETTERM can pass them to a diagnostic.
                                0000    402 ;
                                0000    403 ;               Added new system service $DS_GETTERM, which returns
                                0000    404 ;               the characteristics of the user terminal.
                                0000    405 ;
                                0000    406 ;       88      Bob Bergazzi    Sep 22, 1983    Version 6.13
                                0000    407 ;               Added a $CANTIM if a ^C is typed to cancel a $SETIMR
                                0000    408 ;               which may have been done at program start for the
                                0000    409 ;               /TIME switch.
                                0000    410 ;               Commented out for version 6.13 and 6.14, and 7.0
                                0000    411 ;
                                0000    412 ;       89      Bob Bergazzi    11-11-83        Version 6.14
```

```
0000   413 ;                Clear the DS$M_CTRLC and DS$M_CTRLO flags before we call
0000   414 ;                a control-c handler. This fixes the problem of terminal
0000   415 ;                output in a ^C handler being suppressed in S/A.
0000   416 ;
0000   417 ;        90      Bob Bergazzi    12-28-83        Version 6.14
0000   418 ;                The definition of JIB$L_ARB in the JIB was removed from VMS V4,
0000   419 ;                so define it to its old offset value (the space in the JIB is
0000   420 ;                still there).
0000   421 ;
0000   422 ;        91      Bob Bergazzi    2-14-84         Version 6.14
0000   423 ;                Moved the code which builds ptables for the console load device
0000   424 ;                and the boot device into the CONFIG module, called by
0000   425 ;                JSB INI$LOAD_DEVICE. This was done so that SET MEM could call it.
0000   426 ;
0000   427 ;        92      Richard Brown   May 14, 1984    Version 7.0
0000   428 ;                Added display of legal statement on startup.
0000   429 ;
0000   430 ;        93      M. Baggett      May-17-1984     Version 7.0
0000   431 ;                Allow word length unit to be save in RPB.
0000   432 ;
0000   433 ;        94      Bob Bergazzi    May 18, 1984    Version 7.0
0000   434 ;                Made legal display not print out when running under APT.
0000   435 ;
0000   436 ;        95      Bob Bergazzi    June 19, 1984   Version 7.0
0000   437 ;                Also, do not display legal message when CRD is running.
0000   438 ;
0000   439 ;        96      R.E. Muse       July 13, 1984   Version 7.0
0000   440 ;                Search order bit mask added to $TRNLOG system service.
0000   441 ;                SYS$DISK is now search for in the system logical name
0000   442 ;                table exclusively.
0000   443 ;
0000   444 ;        97      R.E. Muse       19 July 1984    version 7.0
0000   445 ;                logical translation for SYS$SYSROOT added and SYS$DISK
0000   446 ;                removed. Now default directory is "[sys0.sysmaint].
0000   447
0000   448 ;--
```

```
                                0000   450          .Subtitle       Libraries and Macros
                                0000   451 ;
                                0000   452 ; INCLUDE FILES:
                                0000   453 ;
                                0000   454          .Library        /SYS$LIBRARY:LIB/        ;                                  [54]
                                0000   455          .Library        /$CRD/                   ; CRD-specific definitions         [59]
                                0000   456          .Library        /$DS/                    ;                                  [44]
                                0000   457          .Library        /$DIAG/                  ;                                  [44]
                                0000   458
                                0000   459 ;
                                0000   460 ; WEAK EXTERNALS
                                0000   461 ;
                                0000   462
                                0000   463          .WEAK   XDELBPT,XDELTBIT
                                C000   464
                                0000   465 ;
                                0000   466 ; EQUATED SYMBOLS:
                                0000   467 ;
                                0000   468
                    00000001    0000   469 SS$_NORMAL=1                             ; SUCCESS
                                0000   470          $DS_DSDEF                       ; SHOULD PRECEDE ''IOC$K_DIAGPID''
                    00660000    0000   471 IOC$K_DIAGPID == DS$K_SUBSYS@16          ; DIAGNOSTIC PROCESS I.D.
                                0000   472          $DS_DSADEF
                                0000   473          $DS_HPODEF                      ; P-table offsets
                                0000   474          $CCBDEF
                                0000   475          $ARBDEF
                                0000   476          $PHDDEF
                                0000   477          $DIBDEF
                                0000   478          $DEVDEF                         ; Define device type codes
                                0000   479          $PCBDEF
                                0000   480          $JIBDEF
                    00000000    0000   481                  JIB$L_ARB = 0           ; This was removed from VMS v4.0   [90]
                                0000   482          $PRDEF
                                0000   483          $PSLDEF
                                0000   484          $SHRDEF                         ; Define shareable status codes
                                0000   485          $SYIDEF                         ; Define SYI codes                 [49]
                                0000   486          $DS_SCBDEF
                                0000   487          CLIDEF                          ; Define CLI data base offsets     [52]
                                0000   488          CMKDEF
                                0000   489          DSFDEF
                                0000   490          ENVDEF
                                0000   491          $RPBDEF                         ; Define offsets in RPB
                                0000   492          DSQA                            ; QA routine name constants        [44]
                                0000   493          $DS_TYPEDEF                     ; Define type codes                [57]
                                0000   494          $CRD_Literals                   ; Define the CRD literals          [59]
```

```
                     0000   496          .Subtitle       Equated Symbols
                     0000   497
0000000F             0000   498 RPB$V_AutoTest = 15                     ; The AutoTest bit in R5 - take this out[59]
                     0000   499                                          ; when the bit gets defined in $RPBDEF  [59]
                     0000   500                                          ; in Lib.Mlb.                           [59]
                     0000   501
00000000             0000   502 RPB$V_MenuTest = 0                      ; The MenuTest bit in R5 - tkae this out[69]
                     0000   503                                          ; when (and if) the bit gets defined in [69]
                     0000   504                                          ; $RpbDef in Lib.MLb.                   [69]
                     0000   505
00003000             0000   506 DSA$M_DFLTFLGS  == DSA$M_OPER ! DSA$M_PROMPT
                     0000   507
00000200             0000   508 DS$A_PRGBGN     == ^X200
0000F800             0000   509 DS$K_PRGSIZ     == ^XFA00 - DS$A_PRGBGN
                     0000   510
0000000D             0000   511         CR      = 13
0000000A             0000   512         LF      = 10
                     0000   513
                     0000   514 ; INITIAL FREE-CORE REQUEST SIZE
                     0000   515
00000098             0000   516 UCB$SIZE        =       ^X98
00000018             0000   517 DDB$SIZE        =       ^X18
0000002C             0000   518 CRB$SIZE        =       ^X2C
00000020             0000   519 IDB$SIZE        =       ^X20
00000050             0000   520 IRP$SIZE        =       ^X50
00000010             0000   521 CCB$SIZE        =       ^X10
00000050             0000   522 VCB$SIZE        =       ^X50
00000030             0000   523 WCB$SIZE        =       ^X30
00000030             0000   524 FCB$SIZE        =       ^X30
                     0000   525
00000040             0000   526 SGN$C_IRPCNT    ==      64
                     0000   527 SGN$C_NPAGEDYN  ==      <UCB$SIZE*24>           -; 24 UCB'S
                     0000   528                         +<DDB$SIZE*5>           -;  5 DDB'S
                     0000   529                         +<CRB$SIZE*3>           -;  3 CRB'S
                     0000   530                         +<IDB$SIZE*5>           -;  5 IDB'S
                     0000   531                         +<IRP$SIZE*SGN$C_IRPCNT>-; IRP'S
                     0000   532                         +<CCB$SIZE*16>          -; 16 CCB'S
                     0000   533                         +<VCB$SIZE*2>           -;  2 VCB'S
                     0000   534                         +<WCB$SIZE*2>           -;  2 WCB'S
0000263C             0000   535                         +<FCB$SIZE*2>           ;  2 FCB'S
00000800             0000   536 DB_SIZ  = ^X800                      ; Size of DBDRIVER
00000800             0000   537 DL_SIZ  = ^X800                      ; Size of DLDRIVER
00000A00             0000   538 DM_SIZ  = ^XA00                      ; Size of DMDRIVER
00000800             0000   539 DR_SIZ  = ^X800                      ; Size of DRDRIVER
00001000             0000   540 TM_SIZ  = ^X1000                     ; Size of TMDRIVER
00000600             0000   541 XF_SIZ  = ^X600                      ; Size of XFDRIVER
                     0000   542
00005E3C             0000   543 QIO$MINCORE==SGN$C_NPAGEDYN + <DB_SIZ+DL_SIZ+DM_SIZ+DR_SIZ+TM_SIZ+XF_SIZ>
                     0000   544
                     0000   545 ;
                     0000   546 ; ALLOCATION GRANULARITY MASK
                     0000   547 ;
                     0000   548
0000000F             0000   549 MASK     = ^XF
                     0000   550
                     0000   551 ;
                     0000   552 ; STARLET VECTOR LOCAL DEFINITION OVERRIDES DS STARLET SERVICE VECTOR.
```

```
                    0000    553 ;
                    0000    554
800000C8            0000    555 SYS$CRETVA      = ^X800000C8
800000F0            0000    556 SYS$DXLEXH      = ^X800000F0
80000110            0000    557 SYS$DELTVA      = ^X80000110
80000208            0000    558 SYS$SETEXV      = ^X80000208
80000230            0000    559 SYS$SETPRT      = ^X80000230
```

```
                                    0000   561          .Subtitle       Data Psect Declarations
                                00000000   562          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                    0000   563 ;
                                    0000   564 ; Read only data
                                    0000   565 ;
                                    0000   566
                                    0000   567 T_WRONGCPU:
44 49 20 6D 65 74 73 79 53 07 07 00' 0000   568          .ASCIC  <7><7>"System ID register type = !UB, expected !UB!/"   ;       [44]
79 74 20 72 65 74 73 69 67 65 72 20  000C
78 65 20 2C 42 55 21 20 3D 20 65 70  0018
2F 21 42 55 21 20 64 65 74 63 65 70  0024
                              2F     0000
                                    0030   569
                                    0030   570 T_Both_CRD_Bits_Set:                                                ;       [75]
4E 49 20 52 4F 52 52 45 20 3F 3F 00' 0030   571          .Ascic  "?? ERROR IN STARTING CRD - INCORRECT BITS SET!/"   ;       [75]
52 43 20 47 4E 49 54 52 41 54 53 20  003C
43 45 52 52 4F 43 4E 49 20 2D 20 44  0048
2F 21 54 45 53 20 53 54 49 42 20 54  0054
                              2F     0030
                                    0060   572
                                    0060   573 DS$GT_LEGAL:                                                      ;       [92]
58 41 56 5F 21 5F 21 2F 21 2F 21 00' 0060   574          .ASCIC  "!/!/!_!_VAX DIAGNOSTIC SOFTWARE" -                 ;       [92]
20 43 49 54 53 4F 4E 47 41 49 44 20  006C
                  45 52 41 57 54 46  0078
50 4F 52 50 5F 21 5F 21 5F 21 2F 21  0080   575                  "!/!_!_!_PROPERTY OF" -                         ;       [92]
                        46 4F 20 59  008C
49 44 20 20 20 20 20 20 5F 21 2F 21  0093   576                  "!/!_       DIGITAL EQUIPMENT CORPORATION" -    ;       [92]
4D 50 49 55 51 45 20 4C 41 54 49 47  009F
54 41 52 4F 50 52 4F 43 20 54 4E 45  00AB
                              4E 4F  00B7
2A 2A 20 20 20 20 5F 21 2F 21 2F 21  00BA   577                  "!/!/!_     ***CONFIDENTIAL AND PROPRIETARY***" -   ;
41 49 54 4E 45 44 49 46 4E 4F 43 2A  00C6
49 52 50 4F 52 50 20 44 4E 41 20 4C  00D2
                  2A 2A 2A 59 52 41  00DE
75 41 20 65 73 55 2F 21 2F 21 2F 21  00E6   578                  "!/!/!/Use Authorized Only Pursuant to a Valid Right-to-Use License!
6C 6E 4F 20 64 65 7A 09 72 6F 68 74  00F2
74 20 74 6E 61 75 73 72 75 50 20 79  00FE
69 52 20 64 69 6C 61 56 20 61 20 6F  010A
4C 20 65 73 55 2D 6F 74 2D 74 68 67  0116
      2F 21 2F 21 65 73 6E 65 63 69  0122
                              CB     0060
```

```
                                012C   580          .Subtitle       Work Psect Declarations
                           00000000    581          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                0000   582
                                0000   583 DS$FAB_INPUT::
                                0000   584          $FAB    FAC=GET, FNA=SYS$INPUT+1, FNS=9
                                0050   585
                                0050   586 DS$RAB_INPUT::
                                0050   587          $RAB    FAB=DS$FAB_INPUT, ROP=CVT
                                0094   588
                                0094   589 DS$FAB_OUTPUT::
                                0094   590          $FAB    FAC=PUT, FNA=SYS$OUTPUT+1, FNS=10, FOP=CIF
                                00E4   591
                                00E4   592 DS$RAB_OUTPUT::
                                00E4   593          $RAB    FAB=DS$FAB_OUTPUT, ROP=CCO
                                0128   594
                                0128   595
                                0128   596 DS$GA_DS_Ctrl_C_First::                ; Address of 1st DS Control-C Handler   [70]
                    00000000    0128   597          .Long   0                    ; No handler by default.                [51]
                                012C   598
                                012C   599 DS$GA_DS_Ctrl_C_Second::               ; Address of 2nd DS Control-C handler   [70]
                    00000000    012C   600          .Long   0                    ; No handler by default.                [70]
                                0130   601
                                0130   602 DS$L_UserCntrlC::                     ;                                       [73]
                    00000000    0130   603          .Long   0                    ; Address of user Control-C handler
                                0134   604
                                0134   605 EXIT$DESBLK:                         ; DESCRIPTOR OF EXIT HANDLER
                    00000000    0134   606          .LONG   0                    ; FORWARD LINK
                    00000A64'   0138   607          .ADDRESS        EXIT$HANDLR  ; ADDRESS OF EXIT HANDLER
                    00000001    013C   608          .LONG   1                    ; ONE PARAMETER
                    00000144'   0140   609          .ADDRESS        10$          ; ADDRESS TO STORE EXIT REASON
                    00000000    0144   610 10$:     .LONG   0
                                0148   611
                    0000014C    0148   612 SAVEFP: .BLKL    1                    ; INITIAL CONTENTS OF FP
                                014C   613
                    00000174    014C   614 DS$AT_DDIR:     .BLKW   20           ; Saved default directory
                                0174   615
                    0000019C    0174   616 DS$AT_DDEV:     .BLKW   20           ; Saved default device
                                019C   617
                    0000039C    019C   618 IMAGE_HEADER::  .BLKB   512          ; Storage for diagnostic's image header
                                039C   619 IMAGE_HEADER_END::                    ; End of buffer.
```

```
                                039C   621
                                039C   622 ;
                                039C   623 ; PERMANENT CCB DATA BASE
                                039C   624 ;
                     00000040   039C   625 DS$K_CCB          ==      64                  ; NUMBER OF CCB'S
                                039C   626
                                039C   627 CTL$GL_CCB::
                     0000079C   039C   628                   .BLKB   DS$K_CCB * CCB$C_LENGTH
                                079C   629
                                079C   630 CTL$GL_CCBBASE::
                     0000079C'  079C   631                   .ADDRESS        CTL$GL_CCBBASE
                                07A0   632
                                07A0   633 SCH$GL_CURPCB::              ; And Incidentially the current PCB address
                                07A0   634 DS$GL_PCBVEC::               ; List of PCB's (indexed) (DS has only 1)
                     000002E4'  07A0   635                   .ADDRESS DS$AX_SOFTPCB          ; Address of software PCB
                                07A4   636
                                07A4   637 ;
                                07A4   638 ; FORK QUEUE LISTHEADS
                                07A4   639 ;
                                07A4   640
                                07A4   641 ;               .ALIGN  QUAD
                                07A4   642 SWI$GL_FQFL::                                ; FORWARD LINK
                     000007A4'  07A4   643 A:              .LONG   A                       ; IPL-6 LISTHEAD
                                07A8   644
                                07A8   645 SWI$GL_FQBL::                                ; BACKWARD LINK
                     000007A4'  07A8   646                   .LONG   A                       ;
          000007AC'000007AC'    07AC   647 2$:              .LONG   2$,2$                   ; IPL-7 LISTHEAD
          000007B4'000007B4'    07B4   648 3$:              .LONG   3$,3$                   ; IPL-8 LISTHEAD
          000007BC'000007BC'    07BC   649 4$:              .LONG   4$,4$                   ; IPL-9 LISTHEAD
          000007C4'000007C4'    07C4   650 5$:              .LONG   5$,5$                   ; IPL-10 LISTHEAD
          000007CC'000007CC'    07CC   651 6$:              .LONG   6$,6$                   ; IPL-11 LISTHEAD
                                07D4   652
                     000007D8   07D4   653 BOOT$L_ADP::     .BLKL   1                       ; Address of BOOT adapter               [91]
                     000007DC   07D8   654 BOOT$L_CSR::     .BLKL   1                       ; Address of BOOT device csr            [91]
                     000007E0   07DC   655 BOOT$L_UNIT::    .BLKL   1                       ; Unit number of Boot device            [91]
                     000007E4   07E0   656 BOOT$L_R2::      .BLKL   1                       ; CI port station to boot from      [86][91]
                     000007E8   07E4   657 Boot$L_R5::      .BLKL   1                       ; R5 with switches from VMB/DIAGBOOT    [71]
                     00000007EC 07E8   658 BOOT$L_DEVTYP::  .BLKL   1                       ; device type of BOOT media:            [91]
                                07EC   659                                                  ; 0=massbus,1=RK06/07,2=RL01/02
                                07EC   660 DS$GQ_EFL::                                  ; EVENT FLAGS, LOGICAL.
                     000007F0   07EC   661 DS$GL_EFC0::     .BLKL   1                       ; EVENT FLAG CLUSTER 0.
                     000007F4   07F0   662 DS$GL_EFC1::     .BLKL   1                       ; EVENT FLAG CLUSTER 1.
                     000007F8   07F4   663 DS$GA_LASTADR::  .BLKL   1                       ; LAST ADDRESS USED BY SUPERVISOR.
```

```
                        07F8   665 ;
                        07F8   666 ; DYNAMIC EXECUTIVE STORAGE.
                        07F8   667 ;
                        07F8   668
                0000000F 07F8   669 DS$K_ASCIBUF     == 15
                00000200 07F8   670 DS$K_BUFSIZ      == 512
                00000080 07F8   671 DS$K_STRBUF      == 128
                        07F8   672
                000007FC 07F8   673 DS$GL_FLAGS::    .BLKL   1       ; SUPERVISOR INTERNAL FLAGS
                000007FE 07FC   674 DS$GW_TTIN::     .BLKW   1
                00000800 07FE   675 DS$GW_TTOUT::    .BLKW   1
                00000804 0800   676 DS$GL_ERRCNT::   .BLKL   1       ; RUNNING ERROR COUNT.                          [50]
                        0804   677 DS$GL_HARDERR_COUNT::                ;                                          [50]
                00000808 0804   678                  .BLKL   1       ; Summary of Hard Errors                   [50]
                        0808   679 DS$GL_PREPERR_COUNT::               ;                                          [55]
                0000080C 0808   680                  .BLKL   1       ; Summary of Device Peparation Errors      [55]
                        080C   681 DS$GL_SOFTERR_COUNT::               ;                                          [50]
                00000810 080C   682                  .BLKL   1       ; Summary of Soft Errors                   [50]
                        0810   683 DS$GL_DEVERR_COUNT::                ;                                          [50]
                00000814 0810   684                  .BLKL   1       ; Summary of Device Errors                 [50]
                        0814   685 DS$GL_SYSERR_COUNT::                ;                                          [50]
                00000818 0814   686                  .BLKL   1       ; Summary of System Errors                 [50]
                        0818   687 DS$GL_ERRSUP_COUNT::                ;                                          [50]
                0000081C 0818   688                  .BLKL   1       ; Summary of ErrSup Errors                 [50]
                00000820 081C   689 DS$GL_NUMTEST::  .BLKL   1       ; NUMBER OF TESTS IN PROGRAM.
                00000824 0820   690 DS$GL_FSTTEST::  .BLKL   1       ; FIRST TEST TO EXECUTE
                00000828 0824   691 DS$GL_LSTTEST::  .BLKL   1       ; LAST TEST TO EXECUTE
                0000082C 0828   692 DS$GL_SUBTEST::  .BLKL   1       ; SUBTEST NUMBER FOR LOOPING.
                00000830 082C   693 DS$GA_CHKLPPC::  .BLKL   1       ; ADDR OF CURRENT 'CHKLOOP' CALL
                00000834 0830   694 DS$GA_LOOPADR::  .BLKL   1       ; ADDRESS OF CURRENT SUBTEST
                00000838 0834   695 DS$GL_RUNTIM::   .BLKL   1       ; LOCATION TO HOLD ELAPSED RUN TIME
                0000083C 0838   696 DS$GL_TIMESEC::  .BLKL   1       ; CONVERTED RUN TIME IN SECONDS
                00000840 083C   697 DS$GL_WAITIM::   .BLKL   1       ; NUMBER OF MICROSECONDS TO DELAY
                00000844 0840   698 DS$GL_RADIX::    .BLKL   1       ; DEFAULT RADIX.
                00000848 0844   699 DS$GL_SIZE::     .BLKL   1       ; DEFAULT DATA SIZE.
                        0848   700 DS$GQ_BUFQWD::                    ; BUFFER QUADWORD DESCRIPTOR.
                0000084C 0848   701 DS$GL_BUFLEN::   .BLKL   1       ; LENGTH OF CURRENT BUFFER CONTENTS.
                00000850 084C   702 DS$GA_BUFPTR::   .BLKL   1       ; POINTS TO CURRENT POSITION
                00000851 0850   703 DS$GB_BYTEBUF::  .BLKB   1       ; ONE BYTE BUFFER FOR CHK K.B.
                00000852 0851   704 DS$GB_TYPECODE:: .BLKB   1       ; Type code for 'SET BINARY' output        [46]
                        0852   705 DS$GB_INHIBIT_NAMING::              ; Flag byte for ATTACH                     60]
                00000853 0852   706                  .BLKB   1       ;                                          60]
                00000862 0853   707 DS$GT_ASCIBUF::  .BLKB   DS$K_ASCIBUF     ; 15 BYTES FOR NUMERICAL ASCII
                00000A62 0862   708 DS$GT_BUFFER::   .BLKB   DS$K_BUFSIZ      ; INPUT-OUTPUT BUFFER
                00000AE2 0A62   709 DS$GT_STRBUF::   .BLKB   DS$K_STRBUF      ; COUNTED ASCII STRING BUFFER
                00000AEA 0AE2   710 DS$GL_TERMCHAR:: .BLKQ   1       ; Characteristics of user terminal      [87]
```

```
                                0AEA   712          .Subtitle       Data Psect Declarations
                           0000012C    713          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                012C    714
                                012C    715          MODNAM  KERNEL
                                0133    716
                00000000'00000000' 0133    717 SYSVEC:  .LONG   DS$AQ_SYSSRV, DS$AQ_SSEND       ; SYSTEM SERVICE VECTORS.
                                013B    718
                                013B    719 SSPROT:  $CRETVA INADR=SYSVEC
                                014B    720
```

```
                014B   722 SYS$INPUT:
```

```
      54 55 50 4E 49 24 53 59 53 00' 014B   724          .ASCIC  "SYS$INPUT"
                                 09  014B
                                     0155   725 SYS$OUTPUT:
   54 55 50 54 55 4F 24 53 59 53 00' 0155   726          .ASCIC  "SYS$OUTPUT"
                                 0A  0155
                                     0160   727
                0000FE00 0000FE00    0160   728 SUPBUF:  .LONG   DSA$AL_APTMAIL,DSA$AL_APTMAIL ; APT MAILBOX AREA
```

```
                          0168   730
          0000F9FF 00000200  0168   731 PRGBUF: .LONG   DS$A_PRGBGN,DS$A_PRCBGN+DS$K_PRGSIZ-1 ; PROGRAM OVERLAY AREA
```

```
                        0170   733          .SBTTL  SHELL OF STAND ALONE SUPERVISOR MEMORY.
                    00000000   734          .PSECT  _LAST, PAGE
                        0000   735
                        0000   736 AX_BUFF::
                    00000000   0000   737          $$$ = .
                        0000   738 ;
                        0000   739 ; PROCESS CONTROL BLOCK (HARDWARE PCB).
                        0000   740 ;
                        0000   741
                    00000000   0000   742 PHD_BASE          == $$$
                    00000078   0000   743 PCB_BASE          == $$$+PHD$L_PCB
                    00000200   0000   744          $$$ = $$$ + 512                 ; 1 Page for PHD
                        0000   745
                        0000   746 ;
                        0000   747 ; Job Information Block and Access Rights Block
                        0000   748 ;
                        0000   749
                    00000200   0000   750 DS$AX_JIB         == $$$
                    0000026C   0000   751 DS$AX_ARB         == $$$ + JIB$C_LENGTH
                        0000   752
                        0000   753 ;
                        0000   754 ; PROCESS CONTROL BLOCK (SOFTWARE PCB).
                        0000   755 ;
                        0000   756
                    000002E4   0000   757 DS$AX_SOFTPCB     == $$$ + JIB$C_LENGTH + ARB$C_LENGTH
                        0000   758
                        0000   759 ;
                        0000   760 ; PROCESS CONTROL BLOCK (SOFTWARE PCB).
                        0000   761 ;
                        0000   762
                    00000400   0000   763          $$$ = $$$ + 512                 ; 1 Page for JIB, ARB, and PCB
                        0000   764
                        0000   765 ;
                        0000   766 ; SYSTEM CONTROL BLOCK (SCB).
                        0000   767 ;
                        0000   768
                    00000400   0000   769 SCB_BASE          == $$$
                    00000C00   0000   770          $$$ = $$$ + <4*512>             ; MAX 4 pages for SCB
                    00000C00   0000   771 SCB_UNKINT==$$$
                    00001400   0000   772          $$$ = $$$ + <4*512>             ; MAX 4 pages for unexpected handlers   [83]
                        0000   773
                        0000   774 ;
                        0000   775 ; STACK AREAS.
                        000C   776 ;
                        0000   777 ;       Stacks are ordered so that when a mode overflows it's stack
                        0000   778 ;       it will get an access violation because the stack below it
                        0000   779 ;       is protected by memory management. The interrupt stack is
                        0000   780 ;       above the kernel stack so that kernel stack overflows will not
                        0000   781 ;       fall into the interrupt stack.  The pages below the kernel stack
                        0000   782 ;       are NO ACCESS.  See MEMMGT for setup.
                        0000   783 ;--
                        0000   784
                    00001400   0000   785 STACK_BASE        == $$$
                    00001600   0000   786          $$$ = $$$ + <1*512>             ; 1 pages for no-access
                        0000   787
                    00002A00   0000   788          $$$ = $$$ + <10*512>            ; 10 pages for KERNEL stack             [61]
                    00002A00   0000   789 KSTKPTR == $$$
```

```
                        0000   790
00003200  0000   791           $$$ = $$$ + <4*512>          ; 4 pages for INTERRUPT stack          [65]
00003200  0000   792 ISTKPTR == $$$
                        0000   793
00003800  0000   794           $$$ = $$$ + <3*512>          ; 3 pages for EXECUTIVE stack          [43]
00003800  0000   795 ESTKPTR == $$$
                        0000   796
00003E00  0000   797           $$$ = $$$ + <3*512>          ; 3 pages for SUPERVISOR stack         [43]
00003E00  0000   798 SSTKPTR == $$$
                        0000   799
00004400  0000   800           $$$ = $$$ + <3*512>          ; 3 pages for USER stack
00004400  0000   801 USTKPTR == $$$
                        0000   802
00004400  0000   803 P0PT_BASE       == $$$
```

```
                     0000    805          .SBTTL  DIAGNOSTIC SUPERVISOR BOOTSTRAP ROUTINE.
                 00000000    806          .PSECT  CODE, EXE, NOWRT, SHR, LONG
                     0000    807 ;++
                     0000    808 ; FUNCTIONAL DESCRIPTION:
                     0000    809 ;
                     0000    810 ;       INITIAL SETUP ROUTINE FOR STAND ALONE VERSIONS OF THE DIAGNOSTIC
                     0000    811 ;       SUPERVISOR.
                     0000    812 ;
                     0000    813 ; CALLING SEQUENCE:
                     0000    814 ;
                     0000    815 ;       NONE
                     0000    816 ;
                     0000    817 ; INPUT PARAMETERS:
                     0000    818 ;
                     0000    819 ;       NONE
                     0000    820 ;
                     0000    821 ; IMPLICIT INPUTS:
                     0000    822 ;
                     0000    823 ;       NONE
                     0000    824 ;
                     0000    825 ; OUTPUT PARAMETERS:
                     0000    826 ;
                     0000    827 ;       NONE
                     0000    828 ;
                     0000    829 ; IMPLICIT OUTPUTS:
                     0000    830 ;
                     0000    831 ;       NONE
                     0000    832 ;
                     0000    833 ; COMPLETION CODES:
                     0000    834 ;
                     0000    835 ;       NONE
                     0000    836 ;
                     0000    837 ; SIDE EFFECTS:
                     0000    838 ;
                     0000    839 ;       Plenty!
                     0000    840 ;
                     0000    841 ;--
```

```
                                0000   843 ;+
                                0000   844 ; Normal Diagnostic Supervisor (in stand-alone mode) entry from START 10000
                                0000   845 ;-
                                0000   846
                                0000   847 BOOT::
                    12   1F    DA  0000   848          MTPR    #31,#PR$_IPL                    ; RAISE IPL TO 1F
          5E  00003200'8F    D0  0003   849          MOVL    #ISTKPTR, SP                    ; INITIALIZE INTERRUPT STACK
                    52   50    D0  000A   850          MOVL    R0,R2                           ; Save Flag
                50   01   06    78  000D   851          ASHL    #9-3, #1, R0                    ; PAGE OF QUADWORDS COUNT.
          51  0000FE00'EF    DE  0011   852          MOVAL   DSA$AL_APTMAIL, R1              ; START OF APT MAILBOX, ETC.
                        81    7C  0018   853 10$:     CLRQ    (R1)+                           ; CLEAR IT.
                     FB 50    F5  001A   854          SOBGTR  R0, 10$                         ; UNTIL END.
0000FE00'EF    00003000 8F    C8  001D   855          BISL2   #DSA$M_DFLTFLGS, DSA$GL_FLAGS   ; SET DEFAULT FLAGS.
                                0028   856
                000007D4'EF    D5  0028   857          TSTL    BOOT$L_ADP                      ; Check one-shot flag
                        67    12  002E   858          BNEQ    30$                             ; Branch if shot
                    50   3E    DB  0030   859          MFPR    #PR$_SID,R0                     ; Get SID register
                    52   50    D1  0033   860          CMPL    R0,R2                           ; Entered from DIAGBOOT?
                        5F    12  0036   861          BNEQ    30$                             ; Branch if not
        000007D4'EF    5C   AB    D0  0038   862          MOVL    RPB$L_ADPPHY(R11),L^BOOT$L_ADP  ; Save adapter address
        000007D8'EF    54   AB    D0  0040   863          MOVL    RPB$L_CSRPHY(R11),L^BOOT$L_CSR  ; Save device csr address
        000007DC'EF    64   AB    3C  0048   864          MOVZWL  RPB$W_UNIT(R11),L^BOOT$L_UNIT   ; Save boot unit number          [93]
        000007E0'EF    24   AB    D0  0050   865          MOVL    RPB$L_BOOTR2(R11),L^BOOT$L_R2   ; Save CI port number           [86]
        000007E4'EF    30   AB    D0  0058   866          MOVL    RPB$L_BOOTR5(R11),L^BOOT$L_R5   ; Save flags from boot
        000007E8'EF    66   AB    9A  0060   867          MOVZBL  RPB$B_DEVTYP(R11), -
                                0068   868                  L^BOOT$L_DEVTYP                 ; Save device type from boot
                5A   68   AB    9E  0068   869          MovAB   RPB$T_File(R11), R10            ; point to ASCIC file name      [52]
                    59   8A    9A  006C   870          MovZBL  (R10)+, R9                      ; get length/address in R9/R10  [52]
                        26    13  006F   871          BEql    30$                             ; Don't bother if nothing there [52]
                    50   6A    9A  0071   872          MovZBL  (R10), R0                       ; Get first character of name   [52]
                    50   02    80  0074   873          AddB2   #2, R0                          ; Make ] or > for end of dir.   [52]
                6A  59   50    3A  0077   874          LocC    R0, R9, (R10)                   ; Search for end of directory   [52]
                        1A    13  007B   875          BEql    30$                             ; Exit if not found             [52]
                50  51   5A    C3  007D   876          SubL3   R10, R1, R0                     ; Compute length of the         [52]
                        50    D6  0081   877          IncL    R0                              ; .. file name string           [52]
                    51   5A    D0  0083   878          MovL    R10, R1                         ; Get address of string         [52]
            52  00000000'EF    DE  0086   879          MovAL   L^Ds$GL_CliBase, R2             ; Point to CLI data base        [52]
                 08   A2   50    7D  008D   880          MovQ    R0, Cli$Q_File(R2)              ; Set it up for SET LOAD        [52]
                 00000000'EF    16  0091   881          Jsb     L^Dsv$SetLoad                   ; And do the SET LOAD command  [58]
                                0097   882 30$:
```

```
                                0097   884 ;+
                                0097   885 ; APT ENTRY POINT FROM START 10004
                                0097   886 ;-
                                0097   887 APT::
                     12   1F    DA 0097   888         MTPR    #31,#PR$_IPL                    ; RAISE IPL TO HOLD OFF
                                009A   889                                                     ; ANY INTERRUPTS
            5E   00003200'8F    D0 009A   890         MOVL    #ISTKPTR,SP                     ; SET INTERRUPT STACK
                          5D    D4 00A1   891         CLRL    FP                              ; NO PREVIOUS STACK FRAME
          0000FE14'EF     3E    DB 00A3   892         MFPR    #PR$_SID,DSA$GL_SID             ; SID SAVED IN FULL
00000000'EF   0000FE17'EF     90 00AA   893         MOVB    DSA$GL_SID+3, -
                                00B5   894                 EXE$GB_CPUTYPE                  ; Load processor type for BOOTDRIVR
000007F8'EF   00000082 8F     D0 00B5   895         MOVL    #DS$M_CMDFLG ! DS$M_LODFLG, -   ; SET COMMAND AND LOAD MODE FLAGS.
                                00C0   896                   L^DS$GL_FLAGS
          00000840'EF   10    D0 00C0   897         MOVL    #16, L^DS$GL_RADIX              ; SET UP DEFAULT RADIX.
          11   00000000'8F    DA 00C7   898         MTPR    #SCB_IMAGE, #PR$_SCBB           ; SYSTEM CONTROL BLOCK BASE.
          10   00000078'8F    DA 00CE   899         MTPR    #PCB_BASE, #PR$_PCBB            ; PROCESS CONTROL BLOCK BASE.
                                00D5   900 ;
                                00D5   901 ; CLEAR ALL BUFFER AREAS.
                                00D5   902 ;
          50   00000000'EF    DE 00D5   903         MOVAL   AX_BUFF, R0                     ; START OF BUFFERS.
                          80    7C 00DC   904 10$:    CLRQ    (R0)+                           ; CLEAR IT.
                00004400'8F   50    D1 00DE   905         CMPL    R0, #POPT_BASE                  ; CHECK FOR ENOUGH.
                              F5    19 00E5   906         BLSS    10$                             ; LOOP TIL DONE.
                                00E7   907
                                00E7   908 ;
                                00E7   909 ; SYSTEM CONTROL BLOCK INITIALIZATION.
                                00E7   910 ;
          50   00000000'8F    D0 00E7   911         MOVL    #XDELTBIT,R0                    ; Get address of Tbit handler
                          1E    13 00EE   912         BEQL    20$                             ; Branch if no XDELTA linked
          51   00000000'EF    DE 00F0   913         MOVAL   SCB_IMAGE,R1                    ; Point to SCBB
                    28 A1    60 DE 00F7   914         MOVAL   (R0),SCB$L_TBIT(R1)             ; Set tbit vector
    2C A1   00000000'EF      9E 00FB   915         MOVAB   XDELBPT,SCB$L_BREAK(R1)         ; Set breakpoint vector
03 000007E4'EF    02    E1 0103   916         BBC     #RPB$V_INIBPT,BOOT$L_R5,20$     ; Branch if no initial breakpoint
                  00F2    30 010B   917         BSBW    INI$BRK                         ; Initial XDELTA breakpoint
                                010E   918 20$:
                                010E   919 ;
                                010E   920 ; Disable recognition of ^C's for the duration of this stand-alone           [66]
                                010E   921 ; initialization. Note that the KB_Check routine MUST NOT be called before  [66]
                                010E   922 ; this is executed!!!!!                                                     [66]
                                010E   923 ;
                                010E   924
                                010E   925         $DS_CntrlC_S    Disabl=#1               ; Set Disable flag              [66]
                                0119   926
          00000128'EF   D4      0119   927         ClrL    L^DS$GA_DS_Ctrl_C_First         ; Clear the VDS's 1st ^C routine[72]
          0000012C'EF   D4      011F   928         ClrL    L^DS$GA_DS_Ctrl_C_Second        ; Clear the VDS's 2nd ^C routine[72]
          00000130'EF   D4      0125   929         ClrL    L^DS$L_UserCntrlC               ; Clear the User's ^C handler   [72]
                                012B   930
                                012B   931 ;
                                012B   932 ; Check correct processor type for this supervisor
                                012B   933 ;
           50   00'8F    9A    012B   934         MOVZBL  #<DS$GK_SID@-24>,R0              ; Get expected value for type
      50   0000FE17'EF      91 012F   935         CMPB    DSA$GL_SID+3,R0                 ; Same as current?
                    15    13 0136   936         BEQL    30$                             ; Branch if so
                                0138   937         $PRINTI_S       T_WRONGCPU, -
                                0138   938                   DSA$GL_SID+3,R0           ; No, inform the operator
                                014D   939 ;
                                014D   940 ; Do CPU specific stuff
```

```
                                 014D   941 ;
                                 014D   942 30$:    $DS_LOADPCS_S                          ; Load PCS if COMET has a PCS   [78]
                                 0154   943
          01   0000FE17'EF   91  0154   944         CMPB    DSA$GL_SID+3, #1               ; Is it STAR or SUPERSTAR?      [82]
                         0F   12  015B   945         BNEQ    33$                            ; No                            [82]
       07 0000FE14'EF     17   E1  015D   946         BBC     #23, DSA$GL_SID, 33$           ; Yes, If STAR, branch          [82]
          00000009'EF     3F   D0  0165   947         MOVL    #^X0000003F, DSX$WAITUS_USOVR+9 ; SUPERSTAR, change DS$K_USOVR in WA
                                 016C   948
                                 016C   949 33$:    $DS_INITSCB_G                          ; Initialize SCB
                                 0173   950
                                 0173   951 ;
                                 0173   952 ; MEMORY MANAGEMENT INITIALIZATION.
                                 0173   953 ;
                                 0173   954
               00000000'EF       16  0173   955         JSB     L^MAPMEM                       ; Map physical memory
                                 0179   956
          00000000'EF     6E     FA  0179   957         CALLG   (SP),QIO$INITIALIZE            ; SETUP QIO DATABASE
               00000000'EF       7C  0180   958         CLRQ    L^DS$GL_BUFCNT                 ; INIT BUFFER COUNTS, P0 + P1.
               00000008'EF       7C  0186   959         CLRQ    L^DS$GL_BUFCNT+8               ;  SYS AND (UNUSED).
                     38   00     DA  018C   960         MTPR    #0, #PR$_MAPEN                 ;
               00000000'EF       94  018F   961         CLRB    DS$GB_MM_ENB           ; Memory management not enabled
                                 0195   962
                                 0195   963 ;
                                 0195   964 ; INITIALIZE PROCESS CONTROL BLOCK.
                                 0195   965 ;
                                 0195   966
                                 0195   967 ;
                                 0195   968 ; INITIALIZE EXE$GQ_SYSTIME (SYSTEM DATE/TIME QUADWORD)
                                 0195   969 ;
                                 0195   970
00000007'EF    0000000F'EF       D0  0195   971         MOVL    L^DS$GQ_DAYTIM+15, -           ; GET YEAR FROM LINK DATE
                                 01A0   972                   L^DS$GT_NEWYEAR+7
     00000000'EF    0000'8F     3C  01A0   973         MOVZWL  #DS$K_NYSIZ, L^DS$GQ_CURYEAR   ; BUILD A STRING DESCRIPTOR
00000004'EF    00000000'EF       DE  01A9   974         MOVAL   L^DS$GT_NEWYEAR, -
                                 01B4   975                   L^DS$GQ_CURYEAR+4
                                 01B4   976         $BINTIM_S  L^DS$GQ_CURYEAR, L^DS$GQ_CURYEAR
00000000'EF    00000000'EF       7D  01C7   977         MOVQ    L^DS$GQ_CURYEAR, EXE$GQ_SYSTIME ; PRIME SYSTIME WITH THIS YEAR
               00000000'EF       16  01D2   978         JSB     L^CLK$RE_INIT                  ; INIT INTERVAL TIMER AND CLOCK
               00000000'EF       16  01D8   979         JSB     L^DSR$SETIMR_INI               ; INIT THE TIMER QUEUE'S
                                 01DE   980
                                 01DE   981 ;       PUSHL   #0                             ; CLEAR CONTROL-C HANDLER       [66]
                                 01DE   982 ;       CALLS   #1,L^DSX$CNTRLC                                                [66]
                                 01DE   983
               00000000'EF       16  01DE   984         JSB     L^INI$PTABLE                   ; Clear P-table pointers
                                 01E4   985
               0000073B'EF       16  01E4   986         JSB     L^INIT_CONTEXT                 ; Initialize Context....
                                 01EA   987
               00000000'EF       16  01EA   988         JSB     L^INI$LOAD_DEVICE              ; Make ptables for console and  [91]
                                 01F0   989                                                  ; boot devices                  [91]
                                 01F0   990
                                 01F0   991 ; Size the console terminal.                                                [87]
                                 01F0   992
          00000000'EF     00     FB  01F0   993         CALLS   #0,SIZE_TERM                   ;                               [87]
                                 01F7   994
                                 01F7   995 ;
                                 01F7   996 ; Initialize RMS file control block pointer table (RMS$FILE_TABLE)
                                 01F7   997 ;
```

```
                00000000'EF   16  01F7   998          JSB     L^RMS$INIT                      ; Call initialization routine
                       03DC   31  01FD   999          BRW     BEGIN0
                                  0200  1000
                                  0200  1001 INI$BRK::
                              03  0200  1002          BPT                                     ; Known XDELTA breakpoint
                              05  0201  1003          RSB                                     ; Return
                                  0202  1004 DS$SOFTIPL5::
                00000000'8F   D5  0202  1005          TSTL    #XDELBPT                        ; XDELTA present?
                              02  13  0208  1006          BEQL    10$                             ; Branch if not
                              F4  10  020A  1007          BSBB    INI$BRK                         ; Take XDELTA breakpoint
                              02  02  020C  1008 10$:     REI                                     ; Return from IPL 5
```

```
                020D  1010          .SBTTL  USEP KERNEL ROUTINE
           0000020D  1011          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                020D  1012 ;++
                020D  1013 ; FUNCTIONAL DESCRIPTION:
                020D  1014 ;
                020D  1015 ;       Entry from command mode, perform usermode only initialization.
                020D  1016 ;
                020D  1017 ; CALLING SEQUENCE:
                020D  1018 ;
                020D  1019 ;       CALLS   #0,DS$USERMODE
                020D  1020 ;
                020D  1021 ; INPUT PARAMETERS:      NONE
                020D  1022 ;
                020D  1023 ; IMPLICIT INPUTS:       NONE
                020D  1024 ;
                020D  1025 ; OUTPUT PARAMETERS:     NONE
                020D  1026 ;
                020D  1027 ; IMPLICIT OUTPUTS:      NONE
                020D  1028 ;
                020D  1029 ; COMPLETION CODES:      NONE
                020D  1030 ;
                020D  1031 ; SIDE EFFECTS:          NONE
                020D  1032 ;
                020D  1033 ;--
```

```
                          0FFC  020D  1035 .ENTRY  DS$USERMODE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
            000007F8'EF    D4  020F  1036         CLRL    DS$GL_FLAGS                ; Clear flags initially
                               0215  1037         $CRETVA_S  L^SUPBUF                ; GET ACCESS TO APT MAILBOX AREA
                               0226  1038         $CRETVA_S  L^PRGBUF                ; GET ACCESS TO PROGRAM OVERLAY AREA
                               0237  1039         $SETEXV_S #1,COND_DEBUG            ; SET UP EXCEPTION HANDLER FOR DEBUG
         6D    00000000'EF    DE  024A  1040         MOVAL   L^COND_HANDLR,(FP)         ; SET UP SUPERVISOR'S EXCEPTION HANDLER
                               0251  1041         $DELTVA_G       L^SSPROT           ; UNPROTECT SYSTEM SERVICE VECTORS.
                               025C  1042         $CRETVA_G       L^SSPROT           ; RECAPTURE SPACE.
                               0267  1043         $DCLEXH_S EXIT$DESBLK              ; Declare image exit handler
      80000000'EF    0000'8F    28  0274  1044         MOVC3   #DS$K_SSSIZE, -            ; COPY STARLET SYSTEM SERVICE VECTORS.
                  00000000'EF        027D  1045                 ^X80000000, L^DS$AQ_SYSSRV    ; IN ORDER TO GET THE CORRECT
                               0282  1046                                            ; WORD MASK FOR CALLx
                               0282  1047 ;
                               0282  1048 ; SET UP THE COPIED STARLET SYSTEM SERVICE VECTORS TO JMP TO 80000xxx          [77]
                               0282  1049 ; SO WE DON'T HAVE TO WORRY THAT THE SYSTEM SERVICE VECTOR CODE REMAINS IN     [77]
                               0282  1050 ; 3 PAGES (WHICH IS ALL WE HAVE ROOM FOR IN ENTRY)                            [77]
                               0282  1051 ;
         50    80000002'8F    D0  0282  1052         MOVL    #^X80000002,R0             ; BEGINNING OF SYSTEM SERVICE VECTOR    [77]
         51    00000000'EF    DE  0289  1053         MOVAL   DS$AQ_SYSSRV,R1            ; 10200                                 [77]
                               0290  1054                                            ; QIOW                                  [77]
          02 A1    9F17 8F    B0  0290  1055         MOVW    #^X9F17,2(R1)              ; JMP OPCODE, SKIPPING WORD MASK        [77]
             04 A1    50    D0  0296  1056         MOVL    R0,4(R1)                   ; ABSOLUTE ADDRESS 80000002             [77]
                               029A  1057                                            ; CALL_HANDL                            [77]
                50    0E    C0  029A  1058         ADDL2   #^XE,R0                    ; 80000010                              [77]
          10 A1    9F17 8F    B0  029D  1059         MOVW    #^X9F17,^X10(R1)           ; 10210                                 [77]
             12 A1    50    D0  02A3  1060         MOVL    R0,^X12(R1)                ; 80000010                              [77]
                               02A7  1061                                            ; THE REST CAN BE LOOPED                [77]
                51    1A    C0  02A7  1062         ADDL2   #^X1A,R1                   ; START AT 1021A                        [77]
                50    0A    C0  02AA  1063         ADDL2   #^XA,R0                    ; JUMPING TO 8000001A                   [77]
             61    9F17 8F    B0  02AD  1064 2$:     MOVW    #^X9F17,(R1)               ; SET UP THE JMP                        [77]
                02 A1    50    D0  02B2  1065         MOVL    R0,2(R1)                   ; SET UP THE ADDRESS TO JMP TO          [77]
                50    08    C0  02B6  1066         ADDL2   #8,R0                      ; NEXT...                               [77]
FFEA 51    08    00000000'8F    F1  02B9  1067         ACBL    #DS$AQ_SSEND,#8,R1,2$      ; DONE ?                        [77]
                               02C3  1068 ;
         50    0000014B'EF    9E  02C3  1069         MOVAB   L^SYS$INPUT,R0             ; ADDRESS OF LOGICAL NAME
                        02C7  30  02CA  1070         BSBW    TRAN_ASSIGN                ; TRANSLATE AND ASSIGN
                     06 50    E8  02CD  1071         BLBS    R0,100$                    ; Branch if success                     [87]
                00000353'EF    17  02D0  1072         JMP     10$                        ; Jump if failure                       [87]
                               02D6  1073 100$:                                        ;                                       [87]
      000007FC'EF    51    B0  02D6  1074         MOVW    R1,L^DS$GW_TTIN            ; SAVE INPUT CHANNEL NUMBER
                     7E    D5  02DD  1075         TSTL    -(SP)                      ; Allocate space for characteristics
                     7E    7F  02DF  1076         PUSHAQ  -(SP)                      ; Allocate 8 more bytes and push addr
                     0C    DD  02E1  1077         PUSHL   #12                        ; Length of Device characteristics buf
                  50    6E    7E  02E3  1078         MOVAQ   (SP),R0                    ; Point to DIB descriptor
                               02E6  1079         $GETCHN_S R1,(R0),(R0)             ; Get characteristics of SYS$INPUT:
                   5E 50    E9  02F8  1080         BLBC    R0,17$                     ; Branch if fails                       [87]
                      1F    BA  02FB  1081         POPR    #^M<R0,R1,R2,R3,R4>        ; R0=len, R1=adr  R2,R3,R4=DEVchar
                48 52    0E    E1  02FD  1082         BBC     S^#DEV$V_FOD, R2, 5$       ; Branch if not command file
                               0301  1083         $OPEN   DS$FAB_INPUT               ; Open command file
                   42 50    E9  030E  1084         BLBC    R0, 10$                    ; Exit if error
                               0311  1085         $CONNECT        DS$RAB_INPUT       ; Connect record block
                   32 50    E9  031E  1086         BLBC    R0, 10$                    ; Exit if error
                               0321  1087         $CREATE DS$FAB_OUTPUT              ; Open output file
                   22 50    E9  032E  1088         BLBC    R0, 10$                    ; Exit if error
                               0331  1089         $CONNECT        DS$RAB_OUTPUT      ; Connect record block
                   12 50    E9  033E  1090         BLBC    R0, 10$                    ; Exit if error
      000007F8'EF    16    E3  0341  1091         BBCS    S^#DS$V_BATCH, -
```

```
                    6E        0348  1092                    L^DS$GL_FLAGS, 20$      ; Set 'RMS' mode bit & skip
                              0349  1093 5$:
                              0349  1094
          50  00000155'EF  9E  0349  1095          MOVAB    L^SYS$OUTPUT,R0         ; ADDRESS OF LOGICAL NAME FOR OUTPUT
                     0241  30  0350  1096          BSBW     TRAN_ASSIGN             ; TRANSLATE AND ASSIGN
                              0353  1097 10$:
                   08 50  E8  0353  1098          BLBS     R0,15$                  ; Branch if no error
                    0202  31  0356  1099          BRW      DS$USERMODE_X           ; Exit w/ error
                              0359  1100
                5E  14  C0  0359  1101 17$:      ADDL     #20,SP                  ; Fix stack after $GETCHN error      [87]
                    F5  11  035C  1102          BRB      10$                     ; and leave.                         [87]
                              035E  1103 15$:
      000007FE'EF  51  B0  035E  1104          MOVW     R1,L^DS$GW_TTOUT        ; SAVE OUTPUT CHANNEL NUMBER
                              0365  1105
                   7E  D5  0365  1106          TSTL     -(SP)                   ; Allocate space for characteristics [87]
                   7E  7F  0367  1107          PUSHAQ   -(SP)                   ; Allocate 8 more bytes and push addr [87]
                   0C  DD  0369  1108          PUSHL    #12                     ; Length of Device characteristics buf [87]
                50  6E  7E  036B  1109          MOVAQ    (SP),R0                 ; Point to DIB descriptor            [87]
                              036E  1110          $GETCHN_S R1,(R0),(R0)           ; Get characteristics of SYS$OUTPUT: [87]
                 D6 50  E9  0380  1111          BLBC     R0,17$                  ; Branch if fails                    [87]
                   1F  BA  0383  1112          POPR     #^M<R0,R1,R2,R3,R4>     ; R0=len, R1=adr  R2,R3,R4=DEVchar   [87]
      00000AE2'EF  53  7D  0385  1113          MOVQ     R3,DS$GL_TERMCHAR       ; Save terminal characteristics (R3,R4) [87]
                              038C  1114
                50  7E  7E  038C  1115          MOVAQ    -(SP),R0                ; Address of quad status block
                              038F  1116          $QIOW_S  ,L^DS$GW_TTIN,#IO$_SETMODE!IO$M_CTRLCAST,(R0),,,L^RCTRLC
                50  8E  7D  03B4  1117          MOVQ     (SP)+,R0                ; Get status
                              03B7  1118 20$:
                              03B7  1119          $CRETVA_S  L^SUPBUF              ; GET ACCESS TO APT MAILBOX AREA
                              03C8  1120          $CRETVA_S  L^PRGBUF              ; GET ACCESS TO PROGRAM OVERLAY AREA
                              03D9  1121
                              03D9  1122 ;
                              03D9  1123 ;       Translate the current sys$disk and save it
                              03D9  1124 ;       also set default directory and save the current value
                              03D9  1125 ;
                              03D9  1126 ;       This code supports rooted system directories of the form
                              03D9  1127 ;       [SYSx.SYSMAINT].  If SYS$SYSROOT translates (recursively) to
                              03D9  1128 ;       something of the form "_DRA0:[SYS0.]", the root directory name
                              03D9  1129 ;       will be merged into the new default, and SYS$DISK will be
                              03D9  1130 ;       re-defined to exclude the directory portion.
                              03D9  1131 ;
          52  00000174'EF  9E  03D9  1132          MOVAB    L^DS$AT_DDEV, R2        ; Address of save area for sys$disk
                   02 A2  9F  03E0  1133          PUSHAB   2(R2)                   ; Address of area to save sys$disk
                      26  DD  03E3  1134          PUSHL    #38                     ; Length of area
                      06  DD  03E5  1135          PUSHL    #6                      ; logical name table search mask [96]
                      7E  D4  03E7  1136          CLRL     -(SP)                   ; null parameters
                      7E  D4  03E9  1137          CLRL     -(SP)                   ; Null parameter
                   0C AE  7F  03EB  1138          PUSHAQ   12(SP)                  ; Address of buffer to get translation
                      62  3F  03EE  1139          PUSHAW   (R2)                    ; Address to save length of translation [52]
            00000000'EF  7F  03F0  1140          PUSHAQ   SYS$SYSROOT             ; Address of desciptor of "SYS$ROOT" [97]
      00000000'EF  06  FB  03F6  1141          CALLS    #6,SYS$TRNLOG           ; Translate and save current disk
                50  62  3E  03FD  1142          MovAW    (R2), R0                ; Address of length                  [52]
                51  02 A2  9E  0400  1143          MovAB    2(R2), R1               ; Address of string                  [52]
                    0155  30  0404  1144          BsbW     Clean_Translation       ; Clean up translated string        [52]
          56  00000000'EF  9E  0407  1145          MOVAB    DEF$Q_DEV, R6           ; Address of saved default device desc
                   66  62  3C  040E  1146          MOVZWL   (R2), (R6)              ; Set length of string
    04 B6   02 A2  66  28  0411  1147          MOVC3    (R6), 2(R2), @4(R6)     ; Copy device name for show device
                   52  3F  3C  0417  1148          MOVZWL   #63, R2                 ; Max number of iterations
```

```
                                041A  1149
          50   66    01   C3   041A  1150 30$:    SUBL3   #1, (R6), R0              ; Length of string
                     09   15   041E  1151         BLEQ    40$                      ; Branch if one or less characters
          04 B640    3A   91   0420  1152         CMPB    #^A':', @4(R6)[R0]       ; Trailing colon?
                     02   12   0425  1153         BNEQ    40$                      ; Branch if not
                     66   D7   0427  1154         DECL    (R6)                     ; Reduce length
                                0429  1155
                     04   BB   0429  1156 40$:    PushR   #^MR2                    ; Save precious R2 (loop counter)       [52]
               59    66   3C   042B  1157         MovZWL  (R6), R9                 ; Get length of current name string     [52]
             5A   04 B6   9E   042E  1158         MovAB   @4(R6), R10              ; .. and address                        [52]
        6A   59   5B 8F   3A   0432  1159         LocC    #^A"[", R9, (R10)        ; Is there a '[' in the string?         [52]
                     0A   12   0437  1160         BNeq    43$                      ; Yep, do something about it            [52]
          6A   59    3C   3A   0439  1161         LocC    #^A"<", R9, (R10)        ; Otherwise, is there a '<'?            [52]
                     40   13   043D  1162         BEql    48$                      ; Nope--no directory                    [52]
                     02   11   043F  1163         Brb     43$                      ; Do the stuffs                         [52]
                     D7   11   0441  1164 41$:    Brb     30$                      ; Extend the SOBGTR below ('way below)  [52]
               66    50   A2   0443  1165 43$:    SubW2   R0, (R6)                 ; Reduce device length by directory     [52]
               59    50   7D   0446  1166         MovQ    R0, R9                   ; Now R9/R10 describe directory part    [52]
               50    6A   9A   0449  1167         MovZBL  (R10), R0                ; Get directory open character ([ or <) [52]
               50    02   C0   044C  1168         AddL2   #2, R0                   ; Calculate corresponding close char.   [52]
          6A   59    50   3A   044F  1169         LocC    R0, R9, (R10)            ; Find it in the string                 [52]
                     2A   13   0453  1170         BEql    48$                      ; Bomb out if it's not there            [52]
            2E   FF  A1   91   0455  1171         CmpB    B^-1(R1), #^A"."         ; Is previous character a '.'?          [52]
                     24   12   0459  1172         BNeq    48$                      ; Ignore directory if not               [52]
          59   51    5A   C3   045B  1173         SubL3   R10, R1, R9              ; Calculate length just to '.'          [52]
            6A   5B 8F   90   045F  1174         MovB    #^A"[", (R10)            ; Make sure it starts with '[', not '<' [52]
       57   00000000'EF   7E   0463  1175         MovAQ   Def$Q_Dir, R7             ; Get descriptor address of directory   [52]
          58   59    01   C3   046A  1176         SubL3   #1, R9, R8               ; Amount to move name up                [52]
  04 B748   04 B7    67   28   046E  1177         MovC3   (R7), @4(R7), @4(R7)[R8]         ; Move it up                    [52]
     04 B7   6A    59   28   0475  1178         MovC3   R9, (R10), @4(R7)        ; Pre-pend new stuff to it              [52]
             67    59   A0   047A  1179         AddW2   R9, (R7)                 ; Calculate new length                  [52]
                   67   B7   047D  1180         DecW    (R7)                     ;  (first [ was overwritten)            [52]
                   04   BA   047F  1181 48$:    PopR    #^MR2                    ; Restore register R2                   [52]
                04 B6   9F   0481  1182         PUSHAB  @4(R6)                   ; Address of buffer
                   14   DD   0484  1183         PUSHL   #20                      ; Max length of translation
                   7E   7C   0486  1184         CLRQ    -(SP)                    ; Null parameters
                   7E   D4   0488  1185         CLRL    -(SP)                    ; Null parameter
                0C AE   7F   048A  1186         PUSHAQ  12(SP)                   ; Address of desc of buffer for trans
                   66   3F   048D  1187         PUSHAW  (R6)                     ; Store result length here
                   66   7F   048F  1188         PUSHAQ  (R6)                     ; Address of input logical name
    00000000'EF    06   FB   0491  1189         CALLS   #6,SYS$TRNLOG            ; Translate
                   01   BB   0498  1190         PushR   #^MR0                    ; Save status code                      [52]
             50    66   3E   049A  1191         MovAW   (R6), R0                 ; Address of length                     [52]
          51   04 B6   9E   049D  1192         MovAB   @4(R6), R1               ; Address of string                     [52]
                00B8   30   04A1  1193         BsbW    Clean_Translation        ; Removc esc sequence and '__'          [52]
                   01   BA   04A4  1194         PopR    #^MR0                    ; restore status code                   [52]
          5E   08 AE   9E   04A6  1195         MOVAB   8(SP), SP                ; Remove descriptor
               0A 50   E9   04AA  1196         BLBC    R0,50$                   ; Exit if error
       0000'8F    50   B1   04AD  1197         CMPW    R0, #SS$_NOTRAN          ; Translated?
                   03   13   04B2  1198         BEQL    50$                      ; Branch if no translation
               8A 52   F5   04B4  1199         SOBGTR  R2, 41$                  ; Branch in case of recursion           [52]
                                04B7  1200
          50   66    01   C3   04B7  1201 50$:    SUBL3   #1, (R6), R0             ; Length of current device string
          04 B640    3A   91   04BB  1202         CMPB    #^A":", @4(R6)[R0]       ; Trailing ":"
                     09   13   04C0  1203         BEQL    60$                      ; Branch if so
                     50   D6   04C2  1204         INCL    R0                       ; Update length
          04 B640    3A   90   04C4  1205         MOVB    #^A":", @4(R6)[R0]       ; Insert ":"
```

```
                    66   D6  04C9  1206          INCL    (R6)                        ; Set new length
                                04CB  1207
                    7E   D4  04CB  1208 60$:     ClrL    -(Sp)                       ; Access mode                          [53]
          00000000'EF   7F  04CD  1209          PushAQ  Def$Q_Dev                   ; point to descriptor of new name      [53]
          00000000'EF   7F  04D3  1210          PushAQ  SYS$DISK                    ; Address of descriptor of logical name
                    02   DD  04D9  1211          PushL   #2                          ; In process logical name table        [53]
     00000000'EF    04   FB  04DB  1212          CallS   #4, Sys$CreLog              ; Create logical name                  [53]
                                04E2  1213
          0000014E'EF   9F  04E2  1214          PUSHAB  L^DS$AT_DDIR+2              ; Address of area to save directory
                    26   DD  04E8  1215          PUSHL   #38                         ; Length of area
                    6E   7F  04EA  1216          PUSHAQ  (SP)                        ; Address of buffer for old
          0000014C'EF   3F  04EC  1217          PUSHAW  L^DS$AT_DDIR                ; Address for length of buffer
          00000000'EF   7F  04F2  1218          PUSHAQ  DEF$Q_DIR                   ; Address of descriptor of current dir
     00000000'EF    03   FB  04F8  1219          CALLS   #3,SYS$SETDDIR              ; Set new default directory
                                04FF  1220
                                04FF  1221 ;
                                04FF  1222 ; Use the (all-new for V3.0 VMS) $GETSYI service to find the processor type
                                04FF  1223 ; we are actually running on, and use this for the DSA$GL_SID longword.  If
                                04FF  1224 ; the service fails for any reason, do the same thing we always did before, and
                                04FF  1225 ; use the DS$GK_SID value, rather than fail the startup.
                                04FF  1226 ;
                                04FF  1227
                    7E   7C  04FF  1228          clrq    -(sp)                       ; Terminate argument list              [49]
          0000FE14'EF   9F  0501  1229          pushab  L^dsa$gl_sid                ; Address of internal SID location     [49]
               7E  1001 8F  B0  0507  1230          movw    #syi$_sid, -(sp)            ; Push item code for full SID          [49]
                    7E   04  B0  050C  1231          movw    #4, -(sp)                   ; And length of buffer (longword)      [49]
                    50   6E  9E  050F  1232          movab   (sp), r0                    ; Remember where we are                [49]
                                0512  1233          $getsyi_s       itmlst=(r0)         ; Get the information                  [49]
                    5E   10  C0  0525  1234          addl2   #4*4, sp                    ; Clear off argument list              [49]
                      0B 50  E8  0528  1235          blbs    r0, 70$                     ; Continue if successful               [49]
0000FE14'EF    00000000'8F  D0  052B  1236          movl    #ds$gk_sid,dsa$gl_sid       ; Set pseudo SID value                 [49]
                                0536  1237
0000FE00'EF    10003000 8F  C8  0536  1238 70$:     BISL2   #DSA$M_DFLTFLGS ! DSA$M_USER, -
                                0541  1239                    L^DSA$GL_FLAGS          ; SET DEFAULT CONTROL FLAGS
                                0541  1240
          00000000'EF   16  0541  1241          Jsb     L^MEMPOOL$INIT              ; SETUP MEMORY POOL                    [52]
                                0547  1242
                    FAB6' 30  0547  1243          BSBW    INI$PTABLE                  ; Clear P-table pointers
                                054A  1244
     00000840'EF    10   D0  054A  1245          MOVL    #16,L^DS$GL_RADIX           ; DEFAULT FOR EXAM/DEP FUNCTIONS
     00000148'EF    5D   D0  0551  1246          MOVL    FP,L^SAVEFP                 ; PRESERVE CLEAN STACK POINTER
                  0081   31  0558  1247          BRW     BEGIN0                      ; AND START FRESH                      [52]
                                055B  1248
                                055B  1249 DS$USERMODE_X:
                          04  055B  1250          RET                                 ; RETURN
```

```
                          055C  ;
                          055C  ; Clean up translated string by removing the leading 4 byte escape
                          055C  ; sequence (if any) and one of two '_' characters if there are any.
                          055C  ;
                          055C  Clean_Translation:                                     ;                                     [52]
                 3F   BB  055C          PushR    #^M<R0,R1,R2,R3,R4,R5>  ; Save some registers               [52]
        50    00 BE   3C  055E          MovZWL   @(Sp), R0               ; Get length of translation          [52]
        51    04 BE   9E  0562          MovAB    @4(Sp), R1              ; .. and address                     [52]
                 53   D4  0566          ClrL     R3                      ; Length to move it down             [52]
           1B    61   91  0568          CmpB     (R1), #^X1B             ; is it an <ESC>?                    [52]
                 03   12  056B          BNeq     10$                     ; Branch if not                      [52]
           53    04   C0  056D          AddL2    #4, R3                  ; Move down 4 bytes                  [52]
     5F 8F     6143   91  0570  10$:    CmpB     (R1)[R3], #^A'_'        ; Is device preceeded by '__'?       [52]
                 0A   12  0575          BNeq     20$                     ; ..                                 [52]
  5F 8F     01 A143   91  0577          CmpB     1(R1)[R3], #^A'_'       ; ..                                 [52]
                 02   12  057D          BNeq     20$                     ; Branch if not                      [52]
                 53   D6  057F          IncL     R3                      ; Move down one byte (more)          [52]
                 53   D5  0581  20$:    TstL     R3                      ; Do we move any                     [52]
                 0C   13  0583          BEql     30$                     ; Clear out if nothing to move       [52]
           50    53   C2  0585          SubL2    R3, R0                  ; Number of bytes to move            [52]
        00 BE    50   B0  0588          MovW     R0, @(Sp)               ; Set new length of equivalence string [52]
   61    6143    50   28  058C          MovC3    R0, (R1)[R3], (R1)      ; Move 'em out                       [52]
                 3F   BA  0591  30$:    PopR     #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers                  [52]
                      05  0593          Rsb                              ;                                    [52]
                          0594  ;
                          0594  ; TRANSLATE AND ASSIGN CHANNEL
                          0594  ; INPUT:  R0     ADDR OF ASCIC LOGICAL NAME
                          0594  ; OUTPUT: R0     SUCCESS/FAILURE
                          0594  ;         R1     CHANNEL NUMBER
                          0594  TRAN_ASSIGN:
           5E    28   C2  0594          SUBL     #40,SP
                 6E   9F  0597          PUSHAB   (SP)                    ; MAKE DESCRIPTOR OF OUTPUT BUFFER
                 28   DD  0599          PUSHL    #40                     ; LENGTH
                          059B                                           ; ++++ STACK CLEANED TO HERE AFTER CALL
              01 A0   9F  059B          PUSHAB   1(R0)                   ; INPUT DESCRIPTOR
           7E    60   9A  059E          MOVZBL   (R0),-(SP)              ; LENGTH
                 7E   D4  05A1          CLRL     -(SP)                   ; DUMMY ARG
                 7E   7C  05A3          CLRQ     -(SP)                   ; 2 DUMMY ARGS
              14 AE   7F  05A5          PUSHAQ   20(SP)                  ; ADDRESS OF OUTPUT DESCRIPTOR
              18 AE   DF  05A8          PUSHAL   24(SP)                  ; ADDRESS OF OUTPUT LENGTH
              14 AE   7F  05AB          PUSHAQ   20(SP)                  ; ADDRESS OF INPUT DESCRIPTOR
  00000000'EF    08   FB  05AE          CALLS    #8,SYS$TRNLOG           ; TRANSLATE THE NAME
           20    50   E9  05B5          BLBC     R0,120$                 ; BRANCH IF FAILED
                          05B8
     1B       08 AE   91  05B8          CMPB     8(SP),#^X1B             ; ESCAPE?
                 07   12  05BC          BNEQ     110$                    ; NO
           6E    04   C2  05BE          SUBL     #4,(SP)                 ; SHORTEN STRING
        04 AE    04   C0  05C1          ADDL     #4,4(SP)                ; UPDATE ADDRESS
                          05C5  110$:
                 7E   D4  05C5          CLRL     -(SP)                   ; SPACE FOR CHANNEL NUMBER
                 7E   7C  05C7          CLRQ     -(SP)                   ; 2 DUMMY ARGS
              08 AE   DF  05C9          PUSHAL   8(SP)                   ; ADDRESS TO RETURN CHANNEL NUMBER
              10 AE   7F  05CC          PUSHAQ   16(SP)                  ; ADDRESS OF DESCRIPTOR
  00000000'EF    04   FB  05CF          CALLS    #4,SYS$ASSIGN           ; ASSIGN THE CHANNEL
                 02   BA  05D6          POPR     #^MR1                   ; GET CHANNEL ASSIGNED
                          05D8  120$:
           5E    30   C0  05D8          ADDL     #48,SP                  ; CLEANUP STACK
```

```
                        05  05DB  1309          RSB                                 ; RETURN
```

```
                        05DC  1311          .SBTTL  COMMON KERNEL REFRESH ENTRY POINT.
                        05DC  1312 ;++
                        05DC  1313 ; FUNCTIONAL DESCRIPTION:
                        05DC  1314 ;
                        05DC  1315 ;       This routine is the basic loop of the supervisor.
                        05DC  1316 ;       At begin it refreshes the image context and calls DS$CLI.
                        05DC  1317 ;       The diagonostic is entered from the command processor.
                        05DC  1318 ;
                        05DC  1319 ; CALLING SEQUENCE:
                        05DC  1320 ;
                        05DC  1321 ;       ENTERED FROM "BOOT".
                        05DC  1322 ;
                        05DC  1323 ; INPUT PARAMETERS:     NONE
                        05DC  1324 ;
                        05DC  1325 ; IMPLICIT INPUTS:      NONE
                        05DC  1326 ;
                        05DC  1327 ; OUTPUT PARAMETERS:    NONE
                        05DC  1328 ;
                        05DC  1329 ; IMPLICIT OUTPUTS:     NONE
                        05DC  1330 ;
                        05DC  1331 ; COMPLETION CODES:     NONE
                        05DC  1332 ;
                        05DC  1333 ; SIDE EFFECTS:
                        05DC  1334 ;
                        05DC  1335 ;       Cleans stacks, registers, and ends up in Kernel mode.
                        05DC  1336 ;       Call the CRD Image-Loading routine and CRD Initialization if
                        05DC  1337 ;       either of the CRD bits have been set in the DSA flags.
                        05DC  1338 ;--
```

```
                         05DC  1340 BEGIN0:                                  ; Only come here once ...                          [59]
00000000'EF   00    FB   05DC  1341          Calls   #0, L^DSR$Restore_CRD_Vectors  ; Restore the CRD vectors that  [76]
                         05E3  1342                                          ; . . . were originally set up in the   [76]
                         05E3  1343                                          ; . . . DSVECS module.                  [76]
                         05E3  1344
                         05E3  1345          Clear_CRD_MenuTest_Off          ; Clear the bit now                     [80]
                         05EB  1346          Clear_CRD_MenuTest_On           ; Clear the bit now                     [80]
                         05F3  1347          Clear_CRD_AutoTest_Off          ; Clear the bit first                   [80]
                         05FB  1348          Clear_CRD_Trace                 ; Clear the CRD trace bit first         [76]
                         0603  1349
                         0603  1350          ;+                                                                      [69]
                         0603  1351          ; If the AutoTest and the DIAG bits are both set in the R5 passed from  [69]
                         0603  1352          ; DiagBoot, set the CRD_AutoTest bit in the DSA flags and Load CRD and  [69]
                         0603  1353          ; initialize it. If the MenuTest and the DIAG  bits are both set in the [69]
                         0603  1354          ; R5 passed from DiagBoot, set the CRD_MenuTest bit in the DSA flags    [69]
                         0603  1355          ; and Load CRD and initialize it.                                       [69]
                         0603  1356          ;-                                                                      [69]
                         0603  1357
          04FE   30      0603  1358          Bsbw    DSR$Check_AutoTest_Off_Set      ; Check to see if Auto bits set
        29 50    E9      0606  1359          Blbc    R0, 10$                 ; If low bit clear, not AutoTest        [69]
                         0609  1360
          04D4   30      0609  1361          Bsbw    DSR$Check_MenuTest_Off_Set      ; Check to see if Menu bits set also
        19 50    E9      060C  1362          Blbc    R0, 9$                  ; If only Auto set, branch              [75]
                         060F  1363
                         060F  1364          ;+
                         060F  1365          ; Both CRD bits are set - this is a bad error. Print an error message   [75]
                         060F  1366          ; and exit the VDS.                                                     [75]
                         060F  1367          ;-
                         060F  1368
                         060F  1369          $Print  #DS$K_Type_General_Error, -     ; Print an error message . . .  [75]
                         060F  1370                    #DS$K_PrintF, -           ; . . . use PrintF                [75]
                         060F  1371                    L^T_Both_CRD_Bits_Set     ; . . . the format string         [75]
                         0622  1372
00000A37'EF   17         0622  1373          Jmp     L^DSV$Exit                ; Bail out - exit to console      [75]
                         0628  1374
```

```
                         0628 9$:                                         ;                               [75]
                         0628         ;+
                         0628         ; The AutoTest bit is set. This indicates that CRD-AutoTest
                         0628         ; Off-line should be run.                                                  [80]
                         0628         ;-
                         0628
                         0628         Set_CRD_AutoTest_Off              ; Set the CRD bit in the DSA flags      [80]
                0E    11 0630         Brb     30$                       ; Go load in the CRD image              [69]
                         0632
              04AB    30 0632 10$:    Bsbw    DSR$Check_MenuTest_Off_Set      ; See if the correct bits are set
             5A 50    E9 0635         Blbc    R0, 100$                  ; If low bit clear, not MenuTest . . .  [69]
                         0638                                           ; . . . and not AutoTest. Go print the  [69]
                         0638                                           ; VDS header line.                      [69]
                         0638
                         0638         ;+
                         0638         ; The MenuTest bit is set. This indicates that CRD-MenuTest
                         0638         ; Offline  should be run.                                                  [80]
                         0638         ;-
                         0638
                         0638         Set_CRD_MenuTest_Off              ; Set the CRD bit in the DSA flags      [80]
                         0640
                         0640         ;+                                                                         [69]
                         0640         ; Load the CRD-Loadable image file. This is done regardless of whether     [69]
                         0640         ; it is CRD Menu Test or CRD Auto Test.                                    [69]
                         0640         ;-                                                                         [69]
                         0640
    00000000'EF    00    FB 0640 30$: Calls   #0, L^DSR$Load_CRD        ; Load the CRD image file               [59]
                         0647
                         0647         ;+
                         0647         ; Call the main CRD routine and have it initialize itself. This routine
                         0647         ; will return failure when its initialization fails for some reason. It
                         0647         ; will return success otherwise. If it fails, the DS image will exit.
                         0647         ; If it succeeds, the DS will continue normally.
                         0647         ;-
                         0647
                00    DD 0647         PushL   #0                        ; Push 0 as param-4                     [59]
                00    DD 0649         PushL   #0                        ; Push 0 as param-3                     [59]
                00    DD 064B         PushL   #CRD$K_CRD_Initialization ; Push Initialization as param-2        [59]
                00    DD 064D         PushL   #CRD$K_Hook_Point         ; Push Hook_Point as param-1            [59]
    00000000'FF    04    FB 064F      Calls   #4, -                     ; Call indirectly the ...               [59]
                         0656           @L^DS$GA_CRD_DS_Interface ; ... CRD$DS_Interface routine.         [59]
             39 50    E8 0656         BlbS    R0, 100$                  ; If routine returns success, branch    [59]
                         0659
                         0659         ;+
                         0659         ; The init routine failed - print an error message and exit the VDS        [74]
                         0659         ;-
                         0659
                         0659         Clear_Binary                      ; Clear the BINARY flag                 [74]
                         0661         Clear_CRD_MenuTest_Off            ; Clear MenuTest flag                   [80]
                         0669         Clear_CRD_MenuTest_On             ; Clear MenuTest flag                   [80]
                         0671         Clear_CRD_AutoTest_Off            ; Clear AutoTest flag                   [80]
                         0679
                         0679         $Print  #DS$K_Type_General_Error, -       ; Print an error message      [74]
                         0679                 #DS$K_PrintF, -           ; . . . use PrintF                      [74]
                         0679                 L^G[illegible]Menu_Error  ; . . . the format string               [74]
                         068C
    00000A37'EF    16    068C         JSb     DSV$Exit                  ; Exit if routine failed                [59]
```

```
                                0692  1433
            52  00000000'EF   DE  0692  1434 100$:  MovAL   L^DS$GL_CLIBASE, R2      ; Set terminal width to 80              [81]
               1C A2    50 8F   9A  0699  1435         MovZBL  #80, CLI$L_DATA(R2)    ; Set the argument                      [81]
                  00000000'EF   16  069E  1436         JSb     VRSETWIDTH             ; ...                                   [81]
    0000FE00'EF   80010800 8F   D3  06A4  1437         BITL    #DSA$M_CRD_AUTOTEST_OFF! -; Don't display legal statement if   [95]
                                0  06AF  1438                 DSA$M_CRD_MENUTEST_OFF! -; CRD MENU or AUTO or if APT      [95]
                                0  06AF  1439                 DSA$M_APT,L^DSA$GL_FLAGS ;                                 [95]
                          13    12  06AF  1440         BNEQ    150$                   ;                                       [95]
                                0  06B1  1441         $PRINT  #DS$K_TYPE_DS_START, -  ; Display statement from legal dept.   [92]
                                0  06B1  1442                 #DS$K_PRINTF,-         ; ... use PRINTF                        [92]
                                0  06B1  1443                 DS$GT_LEGAL            ; ... format statement                  [92]
                                0  06C4  1444
                                0  06C4  1445 150$:  $Print  #DS$K_TYPE_DS_START,-  ; Print startup message                [56]
                                0  06C4  1446                   #DS$K_PRINTF,-       ; ... use PrintF
                                0  06C4  1447                   L^DS$GT_START,-      ; ... the format string
                                0  06C4  1448                   #0                   ; ... the current time
                                0  06D9  1449
                  00000000'EF   16  06D9  1450         JSB     SCRIPT$INIT            ; Init script flags and pointers        [44]
                          52    D4  06DF  1451         CLRL    R2                     ; START AT EFN ZERO
                                0  06E1  1452
                                0  06E1  1453 200$:  $CLREF_S        R2             ; CLEAR THE EVENT FLAG
                    F3 52     3F  F3  06EA  1454         AOBLEQ  #63, R2, 200$          ; CLEAR ALL 64 FLAGS
                                0  06EE  1455
                                0  06EE  1456         $DS_CntrlC_S    Disabl=#0      ; Re-enable catching of ^C's - clear    [66]
                                0  06F9  1457                                        ; ... the Disable flag.                 [66]
                                0  06F9  1458
    0000FE00'EF    01     1F  EF  06F9  1459         ExTZV   #Dsa$V_Apt, #1, -      ; Extract the APT mode bit              [60]
                          50    0  0701  1460                   L^DSA$GL_Flags, R0   ; .. into R0                            [60]
          00000852'EF     50  90  0702  1461         MovB    R0, -                  ; .. and copy byte into flag            [60]
                                0  0709  1462                   L^DS$GB_Inhibit_Naming;                                    [60]
                                0  0709  1463
                                0  0709  1464         Br_If_Not_APT   Begin          ; Branch around if not in APT-mode      [59]
                                0  0711  1465         $PRINTI_S DS$GT_PROMPT         ; Type prompt on APT start
```

```
                                071E  1467 BEGIN::
                                071E  1468 BEGIN_BLISS::
                          1B    10  071E  1469          BSBB    INIT_CONTEXT            ; REINITIALIZE IMAGE CONTEXT
                          7E    DC  0720  1470          MOVPSL  -(SP)                   ; CURRENT PSL
                7E  00000739'EF   9E  0722  1471          MOVAB   20$,-(SP)               ; CURRENT PC
                        7FFF  8F  BB  0729  1472          PUSHR   #^X7FFF                 ; SAVE R0-R14
                00000000'EF   6E  FA  072D  1473          CALLG   (SP),DS$CLI             ; ENTER COMMAND INTERPRETER
                        7FFF  8F  BA  0734  1474          POPR    #^X7FFF                 ; RESTORE R0-R14
                              02  0738  1475          REI                             ; NOTE: USES PC/PSL SAVED/MODIFIED ABOVE
                                  0739  1476
                          E3    11  0739  1477 20$:     BRB     BEGIN                   ; IF DS$CLI EVER RETURNS DO IT AGAIN
```

```
                        1479          .SBTTL  INIT_CONTEXT    Initialize process context
                        1480 ;++
                        1481 ; FUNCTIONAL DESCRIPTION:
                        1482 ;
                        1483 ;       This routine is called to reinitialize the entire process context.
                        1484 ;       It initializes the contents of the registers and resets the stack
                        1485 ;       pointers.  In stand alone it also resets the SYSTEM, P0, and P1 base
                        1486 ;       and length registers, as well as all the stack pointers in all modes.
                        1487 ;       It does not effect the mapping or availablity of memory or the current
                        1488 ;       state of memory management.
                        1489 ;
                        1490 ; CALLING SEQUENCE:
                        1491 ;
                        1492 ;       BSBW    INIT_CONTEXT
                        1493 ;
                        1494 ; INPUT PARAMETERS:     NONE
                        1495 ;
                        1496 ; IMPLICIT INPUTS:
                        1497 ;
                        1498 ;       Initial contents of system, P0, and P1 base registers, stack pointers
                        1499 ;       etc.
                        1500 ;
                        1501 ; OUTPUT PARAMETERS:    NONE
                        1502 ;
                        1503 ; IMPLICIT OUTPUTS:
                        1504 ;
                        1505 ;       The stack and frame pointer are reset it initial values, all registers
                        1506 ;       are modified.
                        1507 ;
                        1508 ; COMPLETION CODES:     NONE
                        1509 ;
                        1510 ; SIDE EFFECTS:
                        1511 ;
                        1512 ;       The entire process context is refreshed.
                        1513 ;--
```

```
                                 073B  1515 INIT_CONTEXT::
                                 073B  1516          QA_MAIN Kernel_Init_Context     ;                                          [44]
        0000FE00'EF    1C   E1  0744  1517          BBC     #DSA$V_USER, L^DSA$GL_FLAGS -
                          03       074B  1518                  ,10$                    ; BRANCH IF STAND ALONE
                       00F6   31  074C  1519          BRW     INIT_USERMODE           ; DO USER MODE STUFF
                                 074F  1520
                                 074F  1521 10$:
     50   000002E4'EF    9E  074F  1522          MOVAB   DS$AX_SOFTPCB, R0               ; GET BASE ADDRESS.
 60  A0   00660000 8F    D0  0756  1523          MOVL    #IOC$R_DIAGPID, PCB$L_PID(R0)   ; SET UP DIAG PID.
    10 A0    10 A0    DE  075E  1524          MOVAL   PCB$L_ASTQFL(R0), PCB$L_ASTQFL(R0) ; INIT AST DELIVERY QUEUE.
    14 A0    10 A0    DE  0763  1525          MOVAL   PCB$L_ASTQFL(R0), PCB$L_ASTQBL(R0) ;
          0C A0    94  0768  1526          CLRB    PCB$B_ASTACT(R0)                ; Clear AST actives
                                 076B  1527 ;
                                 076B  1528 ; Initialize JIB and ARB
                                 076B  1529 ;
     53   00000200'EF    9E  076B  1530          MOVAB   DS$AX_JIB,R3            ; Point to JIB
      78 A0    53   D0  0772  1531          MOVL    R3,PCB$L_JIB(R0)        ; Make PCB point to JIB
     52   0000026C'EF    9E  0776  1532          MOVAB   DS$AX_ARB,R2            ; Point to ARB
    0084 C0    52   D0  077D  1533          MOVL    R2,PCB$L_ARB(R0)        ; Make PCB point to ARB
         63    52   D0  0782  1534          MOVL    R2,JIB$L_ARB(R3)        ; Make JIB point to ARB
                                 0785  1535
     50   00000000'EF    DE  0785  1536          MOVAL   PHD_BASE, R0                    ; GET BASE POINTER.
          00C0 C0 8ED0  078C  1537          POPL    PHD$L_PC(R0)                    ; SAVE RETURN PC.
     00000841'EF    DC  0791  1538          MOVPSL  PSL_SAVE                        ; GET CURRENT PSL IN R4 [68]
                                 0797  1539          CMK     REFRESH
     04   00003200'8F    DA  07A6  1540          MTPR    #ISTKPTR, #PR$_ISP              ; RESET INTERRUPT STACK.
     5E   00002A00'EF    DE  07AD  1541          MOVAL   KSTKPTR, SP                     ; RESET KERNEL STACK POINTER.
          00   5E    DA  07B4  1542          MTPR    SP, #PR$_KSP                    ; INITIALIZE KERNEL STACK
      78 A0    00   DB  07B7  1543          MFPR    #PR$_KSP, PHD$L_KSP(R0)         ; KERNEL STACK POINTER.
     01   00003800'8F    DA  07BB  1544          MTPR    #ESTKPTR, #PR$_ESP              ; INITIALIZE EXECUTIVE STACK
      7C A0    01   DB  07C2  1545          MFPR    #PR$_ESP, PHD$L_ESP(R0)         ; EXECUTIVE STACK POINTER.
     02   00003E00'8F    DA  07C6  1546          MTPR    #SSTKPTR, #PR$_SSP              ; INITIALIZE SUPERVISOR STACK
    0080 C0    02   DB  07CD  1547          MFPR    #PR$_SSP, PHD$L_SSP(R0)         ; SUPERVISOR STACK POINTER.
     03   00004400'8F    DA  07D2  1548          MTPR    #USTKPTR, #PR$_USP              ; INITIALIZE USER STACK
    0084 C0    03   DB  07D9  1549          MFPR    #PR$_USP, PHD$L_USP(R0)         ; USER STACK POINTER.
                                 07DE  1550
                       5D   D4  07DE  1551          CLRL    FP                              ; NO PREVIOUS STACK FRAME
                                 07EC  1552
     51   00B8 C0    DE  07EC  1553          MOVAL   PHD$L_R12(R0),R1                ; POINT TO REGISTER AREA FOR R12
     61    CCCCCCCC 8F    D0  07E5  1554          MOVL    #^XCCCCCCCC,(R1)                ; STORE INITIAL AP
71   61    11111111 8F    C3  07EC  1555 20$:     SUBL3   #^X11111111,(R1),-(R1)          ; STORE NEXT LOWER REGISTER
                       F6   12  07F4  1556          BNEQ    20$                             ; CONTINUE UNTIL R0 STORED
          00C4 C0    DC  07F6  1557          MOVPSL  PHD$L_PSL(R0)                   ; PROCESSOR STATUS LONGWORD.
    00C8 C0    08   DB  07FA  1558          MFPR    #PR$_P0BR, PHD$L_P0BR(R0)       ; P0 BASE REGISTER.
    00CC C0    09   DB  07FF  1559          MFPR    #PR$_P0LR, PHD$L_P0LRASTL(R0)   ; P0 LENGTH REGISTER.
         13    04   DA  0804  1560          MTPR    #4,#PR$_ASTLVL          ; Set ASTLVL
    00CF C0    04   90  0807  1561          MOVB    #4, PHD$B_ASTLVL(R0)            ; SET NO PENDING AST.
    00D0 C0    0A   DB  080C  1562          MFPR    #PR$_P1BR, PHD$L_P1BR(R0)       ; P1 BASE REGISTER.
    00D4 C0    03   DB  0811  1563          MFPR    #PR$_P1LR, PHD$L_P1LR(R0)       ; P1 LENGTH REGISTER.
0088 C0     12345678 8F    D0  0816  1564          MOVL    #^X12345678, PHD$L_R0(R0)       ; ^X12345678 INTO R0 IMAGE.
         12    00   DA  081F  1565          MTPR    #0,#PR$_IPL                     ; NOW LOWER IPL
                                 0822  1566
     50   00C0 C0    D0  0822  1567          MOVL    PHD$L_PC(R0),R0                 ; GET RETURN ADDRESS
                       29   11  0827  1568          BRB     INIT_COMMON                     ; DO COMMON INITIALIZATION
                                 0829  1569 CMK$REFRESH::
         12    1F   DA  0829  1570          MTPR    #31,#PR$_IPL            ; Set IPL to 31  REI doesn't fault
  04 AE   001F0000 8F    D0  082C  1571          MOVL    #PSL$M_IPL,4(SP)                ; SET TO KERNEL MODE & IPL=31
```

```
                04 00000841'EF   04   E1  0834  1572          BBC     #PSL$V_TBIT,PSL_SAVE,10$        ;IF T-BIT SET IN OLD PSL[68]
                        04 AE    10   C8  083C  1573          BISL    #1@PSL$V_TBIT,4(SP)             ;THEN SET IT IN NEW PSL [68]
                                      0840  1574 10$:                                                ;ELSE DON'T.            [68]
                                   02 0840  1575          REI
                                      0841  1576
                           00000000   0841  1577 PSL_SAVE:       .LONG   0                       ;TEMP. STORAGE FOR PSL  [68]
```

```
                                 0845  1579 INIT_USERMODE:
                          50 8ED0  0845  1580          POPL    R0                              ; GET RETURN ADDRESS
          5D    00000148'EF   D0  0848  1581          MOVL    L^SAVEFP,FP                     ; RESET FRAME POINTER
              5E    5D    D0  084F  1582          MOVL    FP,SP                           ; DITTO FOR STACK POINTER
                                 0852  1583
                                 0852  1584 INIT_COMMON:
                                 0852  1585
               00000851'EF    94  0852  1586          clrb    L^ds$gb_typecode                ; Clear the SET BINARY code     [46]
                          01    BB  0858  1587          PUSHR   #^M<R0>                         ; Save return address
                                 085A  1588          $SETAST_S       #1                      ; Enable AST delivery
                          01    BA  0863  1589          POPR    #^M<R0>                         ; Restore return address
          0000086F'EF    00    FB  0865  1590          CALLS   #0,10$                          ; SETUP DUMMY CALL FRAME
                      FEAF    31  086C  1591          BRW     BEGIN                           ; IN THE UNLIKELY EVENT IT RETURN
                                 086F  1592
                                 086F  1593 10$:     .WORD   0                               ; SAVE NO REGISTERS
                          50    DD  0871  1594          PUSHL   R0                              ; PUT RETURN ADDRESS BACK
          6D    00000000'EF    9E  0873  1595          MOVAB   L^COND_HANDLR,(FP)              ; STACK BASED CONDITION HANDLER
                                 087A  1596
          7E    CCCCCCCC 8F    D0  087A  1597          MOVL    #^XCCCCCCCC,-(SP)               ; SET INITIAL AP
       7E 6E    11111111 8F    C3  0881  1598 20$:     SUBL3   #^X11111111,(SP),-(SP)          ; NEXT LOWER REGISTER
                          F6    12  0889  1599          BNEQ    20$                             ; UNTIL R0
                      1FFF 8F    BA  088B  1600          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
          50    12345678 8F    D0  088F  1601          MOVL    #^X12345678,R0                  ; INITIAL R0
                                 0896  1602
                                 05  0896  1603          RSB                                     ; RETURN ALL REFRESHED
```

```
                                0897  1605          .SBTTL  CONTROL-C AST HANDLER
                                0897  1606 ;++
                                0897  1607 ; FUNCTIONAL DESCRIPTION:
                                0897  1608 ;
                                0897  1609 ;       This is the usermode CONTROL-C handler. It sets
                                0897  1610 ;       DS$V_CTRLC in DS$GL_FLAGS and renables the ^C AST.
                                0897  1611 ;
                                0897  1612 ; CALLING SEQUENCE:
                                0897  1613 ;
                                0897  1614 ;       CALLS   #0,RCTRLC
                                0897  1615 ;
                                0897  1616 ; INPUT PARAMETERS:     NONE
                                0897  1617 ;
                                0897  1618 ; IMPLICIT INPUTS:      NONE
                                C897  1619 ;
                                C897  1620 ; OUTPUT PARAMETERS:    NONE
                                0897  1621 ;
                                0897  1622 ; IMPLICIT OUTPUTS:     NONE
                                0897  1623 ;
                                0897  1624 ; COMPLETION CODES:     NONE
                                0897  1625 ;
                                0897  1626 ; SIDE EFFECTS:
                                0897  1627 ;
                                0897  1628 ;       SETS DS$V_CTRLC IN DS$GL_FLAGS
                                0897  1629 ;--
                                0897  1630
                          0000  0897  1631 .ENTRY  RCTRLC,^M<>
5E  00000074 8F     C2        0899  1632          SUBL    #DIB$K_LENGTH,SP        ; Space for device characteristics
                    6E  9F    08A0  1633          PUSHAB  (SP)                    ; Make descriptor of DIB
          00000074  8F  DD    08A2  1634          PUSHL   #DIB$K_LENGTH           ; Set length
               50   5E  D0    08A8  1635          MOVL    SP,R0                   ; Save address of data area
                        08AB  1636          $GETCHN_S L^DS$GW_TTIN,(R0),(R0); Get characterictics
                5E  08  C0    08C1  1637          ADDL    #8,SP                   ; Remove descriptor
                  11 50 E8    08C4  1638          BLBS    R0,10$                  ; Branch if ok
                        08C7  1639          ERRSUP_S
           30 6E    14  E1    08D8  1640 10$:     BBC     #DEV$V_MBX, -
                        08DC  1641                  DIB$L_DEVCHAR(SP),30$   ; Branch if not mailbox
                 08 AE  B7    08DC  1642 20$:     DECW    DIB$L_DEVDEPEND(SP)     ; Decrement count of messaged
                    2B  19    08DF  1643          BLSS    30$                     ; Finish
                        08E1  1644          $QIOW_S ,L^DS$GW_TTIN,#IO$_READVBLK!IO$M_NOW,,,,L^DS$GT_BUFFER,#80
                    D0  11    090A  1645          BRB     20$                     ; Try again
                        090C  1646 30$:
                        090C  1647          $QIOW_S ,L^DS$GW_TTIN,#IO$_SETMODE!IO$M_CTRLCAST,,,,RCTRLC
000007F8'EF         01  C8    092E  1648          BISL2   #DS$M_CTRLC,L^DS$GL_FLAGS       ; SET CONTROL-C FLAG
                        0935  1649
                    04        0935  1650          RET                             ; RETURN
```

```
0936  1652          .SBTTL  KEYBOARD CHECK ROUTINE
0936  1653 ;++
0936  1654 ; FUNCTIONAL DESCRIPTION:
0936  1655 ;
0936  1656 ;       This routine is called to check for ^C, if it has been typed
0936  1657 ;       control is passes to command mode by calling DS$CLI.
0936  1658 ;       If Any ASTS are active (S/A only) the call is ignored
0936  1659 ;
0936  1660 ; CALLING SEQUENCE:
0936  1661 ;
0936  1662 ;       BSBW    KB_CHECK
0936  1663 ;
0936  1664 ; INPUT PARAMETERS:
0936  1665 ;
0936  1666 ;       NONE
0936  1667 ;
0936  1668 ; IMPLICIT INPUTS:
0936  1669 ;
0936  1670 ;       NONE
0936  1671 ;
0936  1672 ; OUTPUT PARAMETERS:
0936  1673 ;
0936  1674 ;       NONE
0936  1675 ;
0936  1676 ; IMPLICIT OUTPUTS:
0936  1677 ;
0936  1678 ;       NONE
0936  1679 ;
0936  1680 ; COMPLETION CODES:
0936  1681 ;
0936  1682 ;       NONE
0936  1683 ;
0936  1684 ; SIDE EFFECTS:
0936  1685 ;
0936  1686 ;       NONE
0936  1687 ;--
```

```
                        0936  1689 KB_CHECK::
     0000FE48'EF   D6   0936  1690          INCL    L^DSA$GL_EVENT          ; INCREMENT EVENT COUNTER
                        093C  1691 KB_CHECK_APT::
0000FE00'EF    1C   E0  093C  1692          BBS     #DSA$V_USER, -          ; CHECK FOR USER ENVIRONMENT.
                   11       0943  1693            L^DSA$GL_FLAGS, 10$
     000002F0'EF   95   0944  1694          TSTB    DS$AX_SOFTPCB+PCB$B_ASTACT ; Any ASTS active?
                   03   13  094A  1695          BEQL    5$                      ; Continue if not                       [51]
                 00BC   31  094C  1696          BRW     KB_CHECK_X              ; Exit immediately if so                [51]
                        094F  1697
     00000000'EF   16   094F  1698 5$:     JSB     L^KB_POLL               ; Poll console
                        0955  1699
000007F8'EF    00   E0  0955  1700 10$:    BBS     #DS$V_CTRLC, -          ; CHECK CONTROL-C FLAG
                   03       095C  1701            L^DS$GL_FLAGS, 12$      ; If set, continue on                   [51]
                 00AB   31  095D  1702          BRW     KB_CHECK_X              ; If clear, exit routine                [51]
                        0960  1703
000007F8'EF    18   E1  0960  1704 12$:    BBC     #DS$V_DISABLCC, -       ; Continue if ^C enabled                [70]
                   03       0967  1705            DS$GL_FLAGS, -          ; . . .                                 [51]
                            0968  1706            13$                     ; . . .                                 [70]
                 00A0   31  0968  1707          Brw     KB_Check_X              ; Exit if ^C is disabled                [70]
                        096B  1708
     00000000'EF   16   096B  1709 13$:    JSB     SCRIPT$STOP             ; Stop script input                     [44]
```

```
                                0971  1711          ;+
                                0971  1712          ; Algorithm:                                                          [70]
                                0971  1713          ; 1. Check the 1st DS ^C handler.                                     [70]
                                0971  1714          ; 2. If there is not one, go to 4.                                    [70]
                                0971  1715          ; 3. If there is one, do the following:                               [70]
                                0971  1716          ;       a. Call the 1st DS handler.                                   [70]
                                0971  1717          ;       b. If it returns failure, go to 4.                            [70]
                                0971  1718          ;       c. If it returns success, clear the ^C flag and return.       [70]
                                0971  1719          ; 4. Check the user ^C handler.                                       [70]
                                0971  1720          ; 5. If there is not one, go to 7.                                    [70]
                                0971  1721          ; 6. If there is one, do the following:                               [70]
                                0971  1722          ;       a. Call the user handler.                                     [70]
                                0971  1723          ;       b. If it returns failure, go to 7.                            [70]
                                0971  1724          ;       c. If it returns success, clear the ^C flag and return.       [70]
                                0971  1725          ; 7. Check the 2nd DS ^C handler.                                     [70]
                                0971  1726          ; 8. If there is not one, go to 10.                                   [70]
                                0971  1727          ; 9. If there is one, do the following:                               [70]
                                0971  1728          ;       a. Call the 2nd DS handler.                                   [70]
                                0971  1729          ;       b. If it returns failure, go to 10.                           [70]
                                0971  1730          ;       c. If it returns success, clear the ^C flag and return.       [70]
                                0971  1731          ; 10. Set up the stack in preparation for doing an REI.               [70]
                                0971  1732          ; 11. Call DS$Cli for VDS command mode.                               [70]
                                0971  1733          ; 12. REI.                                                            [70]
                                0971  1734          ;--                                                                   [70]
                                0971  1735
        00000128'EF    D5       0971  1736          TSTL    L^DS$GA_DS_Ctrl_C_First ; Check the 1st DS Control-C Handler? [70]
                 1F    13       0977  1737          BEQL    15$                     ; If not one, branch                  [70]
     7E 00000128'EF    D0       0979  1738          MOVL    L^DS$GA_DS_Ctrl_C_First,-; Move the 1st handler address       [70]
                                0980  1739            -(SP)                       ; . . . to the stack                  [70]
        00000128'EF    D4       0980  1740          CLRL    L^DS$GA_DS_Ctrl_C_First ; Clear out the 1st handler address  [70]
        00010001 8F    CA       0986  1741          BICL2   #DS$M_CTRLO!DS$M_CTRLC,-; Clear the ^C and ^O bits            [89]
        000007F8'EF             098C  1742            L^DS$GL_FLAGS               ; . . .                               [89]
            9E   6E    FA       0991  1743          CALLG   (SP), @(SP)+            ; Enter the first DS handler          [70]
              01 50    E9       0994  1744          BLBC    R0, 15$                 ; If 1st handler 'failed', check user's [70]
                       05       0997  1745          RSB                             ; . . . and return to the caller.     [70]
                                0998  1746
        00000130'EF    D5       0998  1747 15$:     TSTL    L^DS$L_UserCntrlC       ; ANY USER HANDLER?                   [70]
                 1F    13       099E  1748          BEQL    20$                     ; NO, go check DS's second handler    [70]
     7E 00000130'EF    D0       09A0  1749          MOVL    L^DS$L_UserCntrlC,-(SP) ; GET ADDRESS OF USER CONTROL-C ROUTINE [70]
        00000130'EF    D4       09A7  1750          CLRL    L^DS$L_UserCntrlC       ; CLEAR HANDLER ADDRESS               [70]
000007F8'EF 00010001 8F    CA       09AD  1751          BICL2   #DS$M_CTRLO!DS$M_CTRLC, -       ;                     [89]
                                09B8  1752            L^DS$GL_FLAGS               ; CLEAR CONTROL-C PENDING             [89]
            9E   6E    FA       09B8  1753          CALLG   (SP),@(SP)+             ; ENTER HANDLER
              C1 50    E9       09BB  1754          BLBC    R0, 20$                 ; If handler 'failed', let DS handle
                       05       09BE  1755          RSB                             ; RETURN
                                09BF  1756
        0000012C'EF    D5       09BF  1757 20$:     TstL    L^DS$GA_DS_Ctrl_C_Second; Is there a 2nd DS Control-C Handler? [70]
                 1F    13       09C5  1758          Beql    50$                     ; If not, call DS$Cli.                [70]
     7E 0000012C'EF    D0       09C7  1759          MovL    L^DS$GA_DS_Ctrl_C_Second, -; Move the 2nd handler address    [70]
                                09CE  1760            -(SP)                       ; . . . to the stack                  [70]
        0000012C'EF    D4       09CE  1761          ClrL    L^DS$GA_DS_Ctrl_C_Second; Clear out the 2nd handler address  [70]
        00010001 8F    CA       09D4  1762          BicL2   #DS$M_CtrlO!DS$M_CtrlC,-; Otherwise, clear the ^C and ^O bits  [89]
        000007F8'EF             09DA  1763            L^DS$GL_Flags               ; . . .                               [89]
            9E   6E    FA       09DF  1764          CallG   (SP), @(SP)+            ; Enter the second DS handler         [70]
              B3 50    E9       09E2  1765          BlbC    R0, 15$                 ; If 2nd handler 'failed', call DS$Cli [70]
                       05       09E5  1766          Rsb                             ; . . . and return to the caller.     [70]
                                09E6  1767
```

```
                        09E6  1768 ;50$:    MOVQ    L^BIN_TIME,L^BIN_TIME   ; Was a /TIME value given? (test for 0) [88]
                        09E6  1769 ;        BEQL    60$                     ; Branch if not                         [88]
                        09E6  1770 ;        $GETTIM_S TIMADR=CTRLC_TIME     ; Current time, so CONTINUE /TIME can   [88]
                        09E6  1771 ;                                        ; be calculated.                        [88]
                        09E6  1772 ;        $CANTIM_S REQIDT=#-2            ; Cancel time on /TIME switch.          [88]
                        09E6  1773 ;
                        09E6  1774 50$:
          7E    6E  D0  09E6  1775 60$:     MOVL    (SP),-(SP)              ; COPY RETURN ADDRESS
            04  AE  DC  09E9  1776          MOVPSL  4(SP)                   ; SAVE CURRENT PSL
            08  AE  9F  09EC  1777          PUSHAB  8(SP)                   ; TRUE STACK POINTER AT ENTRY
        3FFF  8F  BB  09EF  1778          PUSHR   #^X3FFF                 ; STACK REGISTERS
00000000'EF   6E  FA  09F3  1779          CALLG   (SP),L^DS$Cli           ; ENTER COMMAND DECODER
                        09FA  1780
                        09FA  1781 ;        MOVQ    L^BIN_TIME,L^BIN_TIME   ; Branch if /TIME not given or /TIME    [88]
                        09FA  1782 ;        BEQL    70$                     ; has expired.                          [88]
                        09FA  1783 ;        SUBL2   L^BEGIN_TIME, -         ; Calculate remaining time for /TIME    [88]
                        09FA  1784 ;                L^CTRLC_TIME            ;                                       [88]
                        09FA  1785 ;        SBWC    L^BEGIN_TIME+4, -       ; This is how to subtract 2 quads       [88]
                        09FA  1786 ;                L^CTRLC_TIME+4          ; Result is time we actually used.      [88]
                        09FA  1787 ;        ADDL2   L^CTRLC_TIME, -         ; Now, calculate how much more to       [88]
                        09FA  1788 ;                L^BIN_TIME              ; let it run                            [88]
                        09FA  1789 ;        ADWC    L^CTRLC_TIME+4, -       ; This adds 2 quads                     [88]
                        09FA  1790 ;                L^BIN_TIME+4            ;                                       [88]
                        09FA  1791 ;        $SETIMR_S EFN=#0, -             ; DS uses efn 0                         [88]
                        09FA  1792 ;                DAYTIM=BIN_TIME, -      ; address of quad with time             [88]
                        09FA  1793 ;                ASTADR=DSI$TIME_AST, -  ; Where to go when done                 [88]
                        09FA  1794 ;                REQIDT=#-2              ; id for /TIME                          [88]
                        09FA  1795 70$:
                        09FA  1796
    50    38  AE  D0  09FA  1797          MOVL    14*4(SP),R0             ; GET NEW STACK POINTER
    70    3C  AE  7D  09FE  1798          MOVQ    15*4(SP),-(R0)          ; PUT NEW PC/PSL THERE
    38 AE     50  D0  0A02  1799          MOVL    R0,14*4(SP)             ; PUT UPDATED NEW STACK POINTER BACK
        7FFF  8F  BA  0A06  1800          POPR    #^X7FFF                 ; RESTORE REGISTERS, CHANGES SP
              02  0A0A  1801          REI                             ; USE NEW PC/PSL
                        0A0B  1802
                        0A0B  1803 KB_CHECK_X:
              05  0A0B  1804          RSB                             ; RETURN
```

```
                                0A0C  1806          .SBTTL  DSX$CNTRLC      Program enable for Control-C intercept
                                0A0C  1807 ;++
                                0A0C  1808 ; FUNCTIONAL DESCRIPTION:
                                0A0C  1809 ;
                                0A0C  1810 ;       This routine enables the diagnostic to intercept the next
                                0A0C  1811 ;       Control-C typed by the operator.
                                0A0C  1812 ;       If a routine has been specified when the next Control-C is
                                0A0C  1813 ;       typed, the enable is cleared and the routine called.
                                0A0C  1814 ;       The enable is effective until a Control-C is typed
                                0A0C  1815 ;       or canceled by calling with a zero routine_address.
                                0A0C  1816 ;
                                0A0C  1817 ;       The routine also allows enabling or disabling control C handling.  If
                                0A0C  1818 ;       the DISABL argument has low bit SET, ^C handling is DISABLED.
                                0A0C  1819 ;
                                CA0C  1820 ; CALLING SEQUENCE:
                                0A0C  1821 ;
                                0A0C  1822 ;       DS$CNTRLC(routine_address)
                                0A0C  1823 ;
                                0A0C  1824 ; INPUT PARAMETERS:
                                0A0C  1825 ;
                                0A0C  1826 ;       4(AP)   Address of routine to call on Control-C
                                0A0C  1827 ;       8(AP)   DISABL flag (LBS to disable, LBC to enable)
                                0A0C  1828 ;
                                0A0C  1829 ; IMPLICIT INPUTS:       NONE
                                0A0C  1830 ;
                                0A0C  1831 ; OUTPUT PARAMETERS:     NONE
                                0A0C  1832 ;
                                0A0C  1833 ; IMPLICIT OUTPUTS:
                                0A0C  1834 ;
                                0A0C  1835 ;       DS$L_USERCNTRLC is set to value of routine_address
                                0A0C  1836 ;
                                0A0C  1837 ; COMPLETION CODES:      NONE
                                0A0C  1838 ;
                                0A0C  1839 ; SIDE EFFECTS:          NONE
                                0A0C  1840 ;--
                                0A0C  1841
                          0000  0A0C  1842 .ENTRY  DSX$CNTRLC,^M<>
                                0A0E  1843
 00000130'EF    04 AC    D0    0A0E  1844          MOVL    4(AP),L^DS$L_USERCNTRLC ; SET USER CONTROL-C ADDRESS
                50       D4    0A16  1845          CLRL    R0
             02 6C       91    0A18  1846          CMPB    (AP),#2                 ; Is DISABL included?
                16       1F    0A1B  1847          BLSSU   10$                     ; No: don't use it
    51  000007F8'EF      9E    0A1D  1848          MOVAB   L^DS$GL_FLAGS, R1       ; Point to internal flags
 50  61  01              18    EF    0A24  1849          EXTZV   #DS$V_DISABLCC,#1,(R1),R0; Extract current value of disable bit
61  01  18  08 AC        F0    0A29  1850          INSV    8(AP),#DS$V_DISABLCC,#1,(R1); Insert new value
        50  50           03    78    0A2F  1851          ASHL    #3,R0,R0                ; Make old value into a
            50           01    88    0A33  1852 10$:     BISB    #1,R0                   ;  WASSET or WASCLR return code
                         04    0A36  1853          RET
```

```
                        1855          .Sbttl  DSV$EXIT, Process EXIT command
                        1856 ;++
                        1857 ; Functional Description:
                        1858 ;
                        1859 ;       This routine is called from CLI when an EXIT command is
                        1860 ;       typed.  It will call DS_CLEANUP, INIT_CONTEXT; then if
                        1861 ;       in standalone, it will HALT; if on-line, it will do a
                        1862 ;       $EXIT call. If the bits in the R5 that was passed from the
                        1863 ;       BOOT routines are setup for CRD-AutoTest, then do some
                        1864 ;       special things.
                        1865 ;
                        1866 ; Inputs:
                        1867 ;
                        1868 ;       None
                        1869 ;
                        1870 ; Outputs:
                        1871 ;
                        1872 ;       None
                        1873 ;
                        1874 ;--
```

```
                                0A37  1876 DSV$EXIT::                                                        ;                                                       [47]
                                0A37  1877          Br_If_Not_LodFlg        10$     ; Branch if no program loaded                                   [63]
                        F5BE'   30    0A3F  1878          Bsbw    Ds_Cleanup              ; Conditionally call program cleanup            [47]
                                      0A42  1879
                        FCF6    30    0A42  1880 10$:     Bsbw    Init_Context            ; Clean up after ourselves                      [47]
                                      0A45  1881
                                      0A45  1882          ;+
                                      0A45  1883          ; Call the Unload_CRD routine - this routine will check to see if CRD   [71]
                                      0A45  1884          ; been running. If it hasn't, it will return. If CRD has been running,  [71]
                                      0A45  1885          ; the routine will deallocate all the memory that was allocated for     [71]
                                      0A45  1886          ; the CRD image. In this case, it will not return to here! It goes to   [71]
                                      0A45  1887          ; either the Begin_Bliss label, or it does a Halt.                      [71]
                                      0A45  1888          ;-
                                      0A45  1889
                          00    DD    0A45  1890          PushL   #0                      ; If the CRD image is in memory, then           [76]
                                      0A47  1891                                          ; . . . don't return to here.                   [76]
                00000000'EF   01    FB    0A47  1892          Calls   #1, DSR$Unload_CRD      ; 'Unload' the CRD image from memory            [76]
                                      0A4E  1893
                                      0A4E  1894          ;+
                                      0A4E  1895          ; If it comes to here, then the CRD image was not in memory.            [76]
                                      0A4E  1896          ;-
                                      0A4E  1897
                                      0A4E  1898          Br_If_User      60$             ; If in User-Mode, don't try to halt            [63]
                                      0A56  1899
                          00    0A56  1900          Halt                                ; Halt processor in standalone                  [63]
                        FCC4    31    0A57  1901          BrW     Begin                   ; ReStart the VDS if console Continue           [71]
                                      0A5A  1902
                                      0A5A  1903          ;+
                                      0A5A  1904          ; We are in User-Mode. Do a system service exit.                        [63]
                                      0A5A  1905          ;-
                                      0A5A  1906
                                      0A5A  1907 60$:     $Exit_S                         ; Exit with success code                        [63]
                            05  0A63  1908          Rsb                             ; (just in case)                                [47]
```

```
                                0A64  1910          .SBTTL  EXIT$HANDLR     Process EXIT handler
                                0A64  1911 ;++
                                0A64  1912 ; FUNCTIONAL DESCRIPTION:
                                0A64  1913 ;
                                0A64  1914 ;       This routine is called from the command interpreter
                                0A64  1915 ;       when the image is forced to exit.  It's sole purpose is
                                0A64  1916 ;       to execute the diagnostic program cleanup code.
                                0A64  1917 ;       DS_CLEANUP is called only if a program is loaded.
                                0A64  1918 ;       In addition any devices allocated by select are deallocated.
                                0A64  1919 ;
                                0A64  1920 ; CALLING SEQUENCE:
                                0A64  1921 ;
                                0A64  1922 ;       EXIT$HANDLR()
                                0A64  1923 ;
                                0A64  1924 ; INPUT PARAMETERS:     NONE
                                0A64  1925 ;
                                0A64  1926 ; IMPLICIT INPUTS:
                                0A64  1927 ;
                                0A64  1928 ;       DS$GL_FLAGS     DS$V_DONFLG, DS$V_LODFLG, DS$V_STRFLG
                                0A64  1929 ;
                                0A64  1930 ; OUTPUT PARAMETERS:    NONE
                                0A64  1931 ;
                                0A64  1932 ; IMPLICIT OUTPUTS:     NONE
                                0A64  1933 ;
                                0A64  1934 ; COMPLETION CODES:     NONE
                                0A64  1935 ;--
                                0A64  1936
                                0A64  1937 EXIT$HANDLR:
                         0FFC   0A64  1938          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
03 000007F8'EF     01     E1   0A66  1939          BBC     #DS$V_LODFLG,L^DS$GL_FLAGS,10$ ; Branch if not program loaded
              F58F'       30   0A6E  1940          BSBW    DS_CLEANUP                ; Conditionally execute program cleanup
                                0A71  1941
   55   00000000'EF          DE   0A71  1942 10$:    MOVAL   DS$GA_PTABLE,R5           ; Point to P-table list
   53   00000000'EF          DE   0A78  1943          MOVAL   DS$GA_SELECTS,R3          ; Point to select bits
             54     65       D0   0A7F  1944          MOVL    (R5),R4                   ; Get max count of units
                                0A82  1945
         52     6544         D0   0A82  1946 20$:    MOVL    (R5)[R4],R2               ; Get address of P-table
                 15          13   0A86  1947          BEQL    30$                       ; Branch if no P-table
      10 0A A2   00          E1   0A88  1948          BBC     #HP$V_ALLOC, -
                                0A8D  1949                  HP$B_FLAGS(R2),30$        ; Branch if should not be deallocated
      0B 0A A2   01          E1   0A8D  1950          BBC     #HP$V_WASALL, -
                                0A92  1951                  HP$B_FLAGS(R2),30$        ; Branch if is not allocated
                                0A92  1952          $DALLOC_S HP$Q_DEVICE(R2)         ; Deallocate device
             E2 54          F5   0A9D  1953 30$:    SOBGTR  R4,20$                    ; Count and branch
                                0AA0  1954 ;
                                0AA0  1955 ; Restore sys$disk and default directory
                                0AA0  1956 ;
   50   00000174'EF          9E   0AA0  1957          MOVAB   L^DS$AT_DDEV,R0           ; Address of new logical name
             02 A0          9F   0AA7  1958          PUSHAB  2(R0)
          7E    60          3C   0AAA  1959          MOVZWL  (R0),-(SP)                ; Length of new logical name
                7E          D4   0AAD  1960          CLRL    -(SP)                     ; Access mode
             04 AE          7F   0AAF  1961          PUSHAQ  4(SP)                     ; point to descriptor of new name
      00000000'EF           7F   0AB2  1962          PUSHAQ  SYS$SYSROOT               ; Address of descriptor of logical name
                02          DD   0AB8  1963          PUSHL   #2                        ; In process logical name table
   00000000'EF  04          FB   0ABA  1964          CALLS   #4,SYS$CRELOG             ; Create logical name
                                0AC1  1965
   50   0000014C'EF          9E   0AC1  1966          MOVAB   L^DS$AT_DDIR,R0           ; Address of old directory
```

```
                04 AE    02 A0    9E   0AC8  1967          MOVAB   2(R0),4(SP)             ; Fill in descriptor
                      6E    60    3C   0ACD  1968          MOVZWL  (R0),(SP)               ; Fill in length
                            7E    7C   0AD0  1969          CLRQ    -(SP)                   ; Null parameters
                      08 AE       7F   0AD2  1970          PUSHAQ  8(SP)                   ; Address of descriptor
        00000000'EF           03    FB   0AD5  1971          CALLS   #3,SYS$SETDDIR          ; Set default directory back
                      5E    08    C0   0ADC  1972          ADDL    #8,SP                   ; Remove extra stuff
                                       0ADF  1973
                                  04   0ADF  1974          RET
```

```
                                0AE0  1976          .SubTitle       DSR$Check_MenuTest_Off_Set Routine
                                0AE0  1977 ;++
                                0AE0  1978 ; FUNCTIONAL DESCRIPTION:
                                0AE0  1979 ;
                                0AE0  1980 ;       This routine checks to see if CRD MenuTest Offline is active.
                                0AE0  1981 ;       It checks the Boot$L_R5 longword and DSA Flags.
                                0AE0  1982 ;       The value in Boot$L_R5 is the value that
                                0AE0  1983 ;       is passes to the VDS from DiagBoot (in R5).
                                0AE0  1984 ;
                                0AE0  1985 ; CALLING SEQUENCE:
                                0AE0  1986 ;
                                0AE0  1987 ;       Jsb/Bsbw/Bsbb DSR$Check_MenuTest_Off_Set
                                0AE0  1988 ;
                                0AE0  1989 ; INPUT PARAMETERS:
                                0AE0  1990 ;
                                0AE0  1991 ;       NONE
                                0AE0  1992 ;
                                0AE0  1993 ; IMPLICIT INPUTS:
                                0AE0  1994 ;
                                0AE0  1995 ;       Boot$L_R5
                                0AE0  1996 ;       DSA$GL_Flags
                                0AE0  1997 ;
                                0AE0  1998 ; OUTPUT PARAMETERS:
                                0AE0  1999 ;
                                0AE0  2000 ;       NONE
                                0AE0  2001 ;
                                0AE0  2002 ; IMPLICIT OUTPUTS:
                                0AE0  2003 ;
                                0AE0  2004 ;       NONE
                                0AE0  2005 ;
                                0AE0  2006 ; COMPLETION CODES:
                                0AE0  2007 ;
                                0AE0  2008 ;       R0 bit 0 set:
                                0AE0  2009 ;               CRD MENU active.
                                0AE0  2010 ;       R0 bit 1 set:
                                0AE0  2011 ;               The correct bits are set  in Boot$L_R5.
                                0AE0  2012 ;               This indicates that CRD MENU
                                0AE0  2013 ;               was invoked from 'T/M' console command. R0 bit
                                0AE0  2014 ;               1 clear indicates that CRD MENU was invoked by VDS
                                0AE0  2015 ;               'CRD' command.
                                0AE0  2016 ;       R0 = 0: CRD MENU not active.
                                0AE0  2017 ;--
                                0AE0  2018 DSR$Check_MenuTest_Off_Set::            ;                                [80]
                                0AE0  2019
                      50   D4   0AE0  2020          Clrl    R0                     ; Assume they are not set        [69]
                                0AE2  2021
0000FE00'EF           10   E1   0AE2  2022          Bbc     #DSA$V_CRD_MenuTest_Off, - ; If the CRD_MenuTest_Off
                           03   0AE9  2023                    L^DSA$GL_Flags,    -  ; ...bit is not set in the DSA flags
                                0AEA  2024                    50$                   ; ... branch to 50$              [84]
                                0AEA  2025
                      50   01   C8   0AEA  2026          BISL2   #1, R0          ; CRD MENU active; set the flag  [84]
                                0AED  2027
                                0AED  2028          ;+                                                             [69]
                                0AED  2029          ; If the DIAG and the MenuTest bits are both set in the R5 passed from  [71]
                                0AED  2030          ; DiagBoot, set the low bit in R0 and return.                  [69]
                                0AED  2031          ;-                                                             [69]
                                0AED  2032
```

```
                000007E4'EF    04   E1  0AED  2033 50$:    Bbc     #Rpb$V_Diag, -          ; If the DIAG bit is clear ...          [79]
                               0E     0AF4  2034            L^Boot$L_R5, -          ; ... in the R5 passed from DiagBoot,   [69]
                                      0AF5  2035            100$                    ; ... branch to 100$.                   [69]
                                      0AF5  2036                                                                            [69]
                                      0AF5  2037        ;+                                                                  [69]
                                      0AF5  2038        ; The DIAG bit is set.                                              [71]
                                      0AF5  2039        ;-                                                                  [69]
                                      0AF5  2040
                000007E4'EF    00   E1  0AF5  2041        Bbc     #Rpb$V_MenuTest, -      ; If the MenuTest bit is clear ...      [69]
                               06     0AFC  2042            L^Boot$L_R5, -          ; ... in the R5 passed from DiagBoot,   [69]
                                      0AFD  2043            100$                    ; ... branch to 100$.                   [69]
                                      0AFD  2044                                                                            [69]
                                      0AFD  2045        ;+                                                                  [69]
                                      0AFD  2046        ; The MenuTest bit is also set. This indicates that CRD-MenuTest    [69]
                                      0AFD  2047        ; should be run.                                                    [84]
                                      0AFD  2048        ;-                                                                  [69]
                                      0AFD  2049
                          50   01   C8  0AFD  2050 75$:    BISL2   #1, R0                  ; CRD MENU active                       [84]
                          50   02   C8  0B00  2051        BISL2   #2, R0                  ; CRD MENU invoked by 'T/M' cosole command
                                      0B03  2052                                        ;                                       [84]
                                      0B03  2053
                               05     0B03  2054 100$:    Rsb                             ; Return to caller                      [69]
```

```
                          0B04  2056          .SubTitle       DSR$Check_AutoTest_Off_Set Routine
                          0B04  2057 ;+
                          0B04  2058 ; FUNCTIONAL DESCRIPTION:
                          0B04  2059 ;
                          0B04  2060 ;       This routine checks to see if CRD AutoTest Offline is active.
                          0B04  2061 ;       It checks the Boot$L_R5 longword or DSA Flags.
                          0B04  2062 ;       The value in Boot$L_R5 is the value that
                          0B04  2063 ;       is passes to the VDS from DiagBoot (in R5).
```

```
                        0B04  2065 ;
                        0B04  2066 ; CALLING SEQUENCE:
                        0B04  2067 ;
                        0B04  2068 ;       Jsb/Bsbw/Bsbb DSR$Check_AutoTest_Off_Set
```

```
                        0B04  2070 ;
                        0B04  2071 ; INPUT PARAMETERS:
                        0B04  2072 ;
                        0B04  2073 ;       NONE
                        0B04  2074 ;
                        0B04  2075 ; IMPLICIT INPUTS:
                        0B04  2076 ;
                        0B04  2077 ;       Boot$L_R5
                        0B04  2078 ;       DSA$GL_Flags
                        0B04  2079 ;
                        0B04  2080 ; OUTPUT PARAMETERS:
                        0B04  2081 ;
                        0B04  2082 ;       NONE
                        0B04  2083 ;
                        0B04  2084 ; IMPLICIT OUTPUTS:
                        0B04  2085 ;
                        0B04  2086 ;       NONE
                        0B04  2087 ;
                        0B04  2088 ; COMPLETION CODES:
                        0B04  2089 ;
                        0B04  2090 ;       R0 = 1: CRD AUTO active
                        0B04  2091 ;       R0 = 0: CRD AUTO not active
                        0B04  2092 ;--
                        0B04  2093 DSR$Check_AutoTest_Off_Set::          ;                                  [80]
                        0B04  2094
                  50 D4 0B04  2095         ClrL    R0                     ; Assume they are not set          [69]
                        0B06  2096
0000FE00'EF       0B E1 0B06  2097         Bbc     #DSA$V_CRD_AutoTest_Off, - ; If the CRD_AutoTest_Off
                  02    0B0D  2098           L^DSA$GL_Flags,    -         ; ... bit is not set in the DSA flags
                        0B0E  2099           50$                          ; ... branch to 50$.               [84]
                        0B0E  2100
                  10 11 0B0E  2101         Brb     75$                    ; The bit is set; set the flag     [79]
                        0B10  2102
                        0B10  2103         ;+                                                              [69]
                        0B10  2104         ; If the AutoTest and the DIAG bits are both set in the R5 passed from [69]
                        0B10  2105         ; DiagBoot, set the low bit in R0 and return.                   [69]
                        0B10  2106         ;-                                                              [69]
                        0B10  2107
000007E4'EF       04 E1 0B10  2108 50$:    Bbc     #Rpb$V_Diag, -         ; If the DIAG bit is clear ...     [79]
                  0A    0B17  2109           L^Boot$L_R5, -               ; ... in the R5 passed from DiagBoot, [69]
                        0B18  2110           100$                         ; ... branch to 100$.              [69]
                        0B18  2111
                        0B18  2112         ;+                                                              [69]
                        0B18  2113         ; The DIAG bit is set.                                          [69]
                        0B18  2114         ;-                                                              [69]
                        0B18  2115
000007E4'EF       0F E1 0B18  2116         Bbc     #Rpb$V_AutoTest, -     ; If the AutoTest bit is clear ... [69]
                  02    0B1F  2117           L^Boot$L_R5, -               ; ... in the R5 passed from DiagBoot, [69]
                        0B20  2118           100$                         ; ... branch to 100$.              [69]
                        0B20  2119
                        0B20  2120         ;+                                                              [69]
                        0B20  2121         ; The AutoTest bit is also set. This indicates that CRD-AutoTest [69]
                        0B20  2122         ; should be run. Return a 1 in R0.                              [69]
                        0B20  2123         ;-                                                              [69]
                        0B20  2124
                  50 D6 0B20  2125 75$:    IncL    R0                     ; CRD AutoTest is running.         [79]
                        0B22  2126
```

```
                    05  0B22  2127 100$:   Rsb                          ; Return to caller                    [69]
                        0B23  2128
```

```
                        0B23  2130 .sbttl DSX$GETTERM  Get User Terminal Characteristics                    [87]
                        0B23  2131
                        0B23  2132 ;++                                                                     [87]
                        0B23  2133 ; FUNCTIONAL DESCRIPTION:                                              [87]
                        0B23  2134 ;                                                                      [87]
                        0B23  2135 ;       This routine simply passes the user terminal's characteristics [87]
                        0B23  2136 ;       to the caller.  The characteristics are stored in bits 0 through[87]
                        0B23  2137 ;       23.  They are defined by the $TTDEF macro of VMS.  The charac- [87]
                        0B23  2138 ;       teristics are determined at startup time and stored in         [87]
                        0B23  2139 ;       DS$GL_TERMCHAR.                                                [87]
                        0B23  2140 ;                                                                      [87]
                        0B23  2141 ; CALLING SEQUENCE:                                                    [87]
                        0B23  2142 ;                                                                      [87]
                        0B23  2143 ;       PUSHAL  addr                                                   [87]
                        0B23  2144 ;       CALLS   #1,DSX$GETTERM                                         [87]
                        0B23  2145 ;                                                                      [87]
                        0B23  2146 ; INPUT PARAMETERS:                                                    [87]
                        0B23  2147 ;                                                                      [87]
                        0B23  2148 ;       4(AP)   Address of longword in which to load characteristics   [87]
                        0B23  2149 ;                                                                      [87]
                        0B23  2150 ; IMPLICIT INPUTS:                                                     [87]
                        0B23  2151 ;                                                                      [87]
                        0B23  2152 ;       DS$GL_TERMCHAR  Longword containing terminal characteristics   [87]
                        0B23  2153 ;                                                                      [87]
                        0B23  2154 ; OUTPUT PARAMETERS:                                                   [87]
                        0B23  2155 ;                                                                      [87]
                        0B23  2156 ;       NONE                                                           [87]
                        0B23  2157 ;                                                                      [87]
                        0B23  2158 ; IMPLICIT OUTPUTS:                                                    [87]
                        0B23  2159 ;                                                                      [87]
                        0B23  2160 ;       NONE                                                           [87]
                        0B23  2161 ;                                                                      [87]
                        0B23  2162 ; COMPLETION CODES:                                                    [87]
                        0B23  2163 ;                                                                      [87]
                        0B23  2164 ;       SS$_NORMAL - Service successfully completed.                   [87]
                        0B23  2165 ;                                                                      [87]
                        0B23  2166 ; SIDE EFFECTS:                                                        [87]
                        0B23  2167 ;                                                                      [87]
                        0B23  2168 ;       NONE                                                           [87]
                        0B23  2169 ;                                                                      [87]
                        0B23  2170
                        0B23  2171
                  0000  0B23  2172 .ENTRY DSX$GETTERM, ^M<>                 ;                           [87]
                        0B25  2173
04 BC  00000AE2'EF  7D  0B25  2174         MOVQ    DS$GL_TERMCHAR, @4(AP)  ; Load term. characteristics [87]
         50    01   D0  0B2D  2175         MOVL    #SS$_NORMAL,R0          ; Indicate successful return. [87]
               04   0B30  2176         RET                             ; Done.                       [87]
                    0B31  2177
                    0B31  2178         .END
```

```
$$$                             = 00004400 R  D  04
$$.TAB                          = 000000E4 R  D  03
$$.TABEND                       = 00000128 R  D  03
$$.TMP                          = 80000000    D
$$.TMP1                         = 00000001    D
$$.TMP2                         = 000000CF
$$ARGS                          = 00000003    D
$$N                             = 00000001    D
$$T1                            = 00000001    D
$ER                             = 00000002    D
$MODULE                           0000012C R  D  02
A                                 000007A4 R  D  03
APT                               00000097 RG D  05
ARB$C_LENGTH                    = 00000078    D
AX_BUFF                           00000000 RG D  04
BEGIN                             0000071E RG D  05
BEGIN0                            000005DC R  D  05
BEGIN_BLISS                       0000071E RG D  05
BIT...                          = 00000003    D
BOOT                              00000000 RG D  05
BOOT$L_ADP                        000007D4 RG D  03
BOOT$L_CSR                        000007D8 RG D  03
BOOT$L_DEVTYP                     000007E8 RG D  03
BOOT$L_R2                         000007E0 RG D  03
BOOT$L_R5                         000007E4 RG D  03
BOOT$L_UNIT                       000007DC RG D  03
CCB$B_AMOD                        00000009    D
CCB$B_STS                         00000008    D
CCB$C_LENGTH                      00000010    D
CCB$K_LENGTH                      00000010    D
CCB$L_DIRP                        0000000C    D
CCB$L_UCB                         00000000    D
CCB$L_WIND                        00000004    D
CCB$SIZE                        = 00000010    D
CCB$W_IOC                         0000000A    D
CLEAN_TRANSLATION                 0000055C R  D  05
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        00000444    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEST                        00000020    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
CLK$RE_INIT                       ********  X   05
CMK$REFRESH                       00000829 RG D  05
CMK$_                           = 00000004    D
CMK$_ASTEXIT                    = 00000000    D
CMK$_CANTIM                     = 0000000B    D
CMK$_HIBER                      = 00000007    D
CMK$_INITSCB                    = 00000008    D
CMK$_LAST                       = 0000000E    D
CMK$_MMENABLE                   = 00000003    D
CMK$_RCONSOLE                   = 00000001    D
CMK$_REFRESH                    = 00000005    D
CMK$_SCHDWK                     = 00000006    D
CMK$_SETIMR                     = 0000000C    D
CMK$_SETIPL                     = 00000009    D
CMK$_SETPRT                     = 0000000D    D
CMK$_SGIPR                      = 0000000A    D
CMK$_TCONSOLE                   = 00000002    D
COND_DEBUG                        ********  X   05
COND_HANDLR                       ********  X   05
CR                              = 0000000D    D
CRB$SIZE                        = 0000002C    D
CRD$K_CRD_INITIALIZATION        = 00000000  G D
CRD$K_DS_CONTROL_C              = 00000001  G D
CRD$K_DS_FLAGS                  = 00000000  G D
CRD$K_FAILURE                   = 00000000    D
CRD$K_HOOK_POINT                = 00000000  G D
CRD$K_LAST_ACCESS_CODE          = 00000002  G D
CRD$K_LAST_DS_VARIABLE          = 00000002  G D
CRD$K_LAST_FUNCTION             = 00000002  G D
```

```
CRD$K_LAST_HOOK_POINT           = 00000001   G D
CRD$K_READ                      = 00000000   G D
CRD$K_SUCCESS                   = 00C00001     D
CRD$K_TYPECODE                  = 00000001   G D
CRD$K_WRITE                     = 00000001   G D
CRETVA$_ACMODE                  = 0000000C     D
CRETVA$_INADR                   = 00000004     D
CRETVA$_NARGS                   = 00000003     D
CRETVA$_RETADR                  = 00000008     D
CTL$GL_CCB                        0000039C RG D   03
CTL$GL_CCBBASE                    0000079C RG D   03
DB_SIZ                          = 00000800     D
DDB$SIZE                        = 00000018     D
DEF$Q_DEV                         ********     X   05
DEF$Q_DIR                         ********     X   05
DEV$V_FOD                       = 0000000E     D
DEV$V_MBX                       = 00000014     D
DIB$K_LENGTH                    = 00000074     D
DIB$L_DEVCHAR                   = 00000000     D
DIB$L_DEVDEPEND                 = 00000008     D
DL_SIZ                          = 00000800     D
DM_SIZ                          = 00000A00     D
DR_SIZ                          = 00000800     D
DS$AQ_SSEND                       ********     X   02
DS$AQ_SYSSRV                      ********     X   02
DS$AT_DDEV                        00000174 R   D   03
DS$AT_DDIR                        0000014C R   D   03
DS$AX_ARB                       = 0000026C RG D   04
DS$AX_JIB                       = 00000200 RG D   04
DS$AX_SOFTPCB                   = 000002E4 RG D   04
DS$A_PRGBGN                     = 00000200   G D
DS$CLI                            ********     X   05
DS$CNTRLC                         ********     X   05
DS$FAB_INPUT                      00000000 RG D   03
DS$FAB_OUTPUT                     00000094 RG D   03
DS$GA_BUFPTR                      0000084C RG D   03
DS$GA_CHKLPPC                     0000082C RG D   03
DS$GA_CRD_DS_INTERFACE            ********     X   05
DS$GA_DS_CTRL_C_FIRST             00000128 RG D   03
DS$GA_DS_CTRL_C_SECOND            0000012C RG D   03
DS$GA_LASTADR                     000007F4 RG D   03
DS$GA_LOOPADR                     00000830 RG D   03
DS$GA_PTABLE                      ********     X   05
DS$GA_SELECTS                     ********     X   05
DS$GB_BYTEBUF                     00000850 RG D   03
DS$GB_INHIBIT_NAMING              00000852 RG D   03
DS$GB_MM_ENB                      ********     X   05
DS$GB_TYPECODE                    00000851 RG D   03
DS$GK_SID                         ********     X   05
DS$GL_BUFCNT                      ********     X   05
DS$GL_BUFLEN                      00000848 RG D   03
DS$GL_CLIBASE                     ********     X   05
DS$GL_DEVERR_COUNT                00000810 RG D   03
DS$GL_EFC0                        000007EC RG D   03
DS$GL_EFC1                        000007F0 RG D   03
DS$GL_ERRCNT                      00000800 RG D   03
DS$GL_ERRSUP_COUNT                00000818 RG D   03
DS$GL_FLAGS                       000007F8 RG D   03
DS$GL_FSTTEST                     00000820 RG D   03
DS$GL_HARDERR_COUNT               00000804 RG D   03
DS$GL_LSTTEST                     00000824 RG D   03
DS$GL_NUMTEST                     0000081C RG D   03
DS$GL_PCBVEC                      000007A0 RG D   03
DS$GL_PREPERR_COUNT               00000808 RG D   03
DS$GL_RADIX                       00000840 RG D   03
DS$GL_RUNTIM                      00000834 RG D   03
DS$GL_SIZE                        00000844 RG D   03
DS$GL_SOFTERR_COUNT               0000080C RG D   03
DS$GL_SUBTEST                     00000828 RG D   03
DS$GL_SYSERR_COUNT                00000814 RG D   03
DS$GL_TERMCHAR                    00000AE2 RG D   03
DS$GL_TIMESEC                     00000838 RG D   03
DS$GL_WAITIM                      0000083C RG D   03
DS$GQ_BUFQWD                      00000848 RG D   03
DS$GQ_CURYEAR                     ********     X   05
DS$GQ_DAYTIM                      ********     X   05
DS$GQ_EFL                         000007EC RG D   03
DS$GT_ASCIBUF                     00000853 RG D   03
DS$GT_BUFFER                      00000862 RG D   03
DS$GT_LEGAL                       00000060 R   D   02
DS$GT_NEWYEAR                     ********     X   05
DS$GT_PROMPT                      ********     X   05
DS$GT_START                       ********     X   05
DS$GT_STRBUF                      00000A62 RG D   03
DS$GW_TTIN                        000007FC RG D   03
DS$GW_TTOUT                       000007FE RG D   03
DS$INITSCB                        ********     X   05
DS$K_ASCIBUF                    = 0000000F   G D
DS$K_BUFSIZ                     = 00000200   G D
DS$K_CCB                        = 00000040   G D
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_NYSIZ                        ********     X   05
DS$K_PRGSIZ                     = 0000F800   G D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SSSIZE                       ********     X   05
DS$K_STRBUF                     = 00000080   G D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
```

```
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_STARI         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$LOADPCS                        ********   X   05
DS$L_USERCNTRLC                   00000130 RG  D   03
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$RAB_INPUT                      00000050 RG  D   03
DS$RAB_OUTPUT                     000000E4 RG  D   03
DS$SOFTIPL5                       00000202 RG  D   05
DS$USERMODE                       0000020D RG  D   05
DS$USERMODE_X                     0000055B R   D   05
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INSFMEM                     = 00660050     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
```

```
DS$_NORMAL                      =  00660001     D
DS$_NOSUPPORT                   =  006600B1     D
DS$_NOTDON                      =  00660030     D
DS$_NOTIMP                      =  006600B0     D
DS$_NULLSTR                     =  00660010     D
DS$_OVERFLOW                    =  00660008     D
DS$_POWER                       =  00660098     D
DS$_PROGERR                     =  00660020     D
DS$_SEVERE                      =  00660004     D
DS$_TRANSL                      =  006600A0     D
DS$_TRUNCATE                    =  00660028     D
DS$_UNEXPINT                    =  006600D8     D
DS$_VASFULL                     =  00660048     D
DS$_WARNING                     =  00660000     D
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSA$M_APT                       =  80000000     D
DSA$M_CRD_AUTOTEST_OFF          =  00000800     D
DSA$M_CRD_MENUTEST_OFF          =  00010000     D
DSA$M_DFLTFLGS                  =  00003000   G D
DSA$M_OPER                      =  00001000     D
DSA$M_PROMPT                    =  00002000     D
DSA$M_USER                      =  10000000     D
DSA$V_APT                       =  0000001F     D
DSA$V_BINARY                    =  00000001     D
DSA$V_CRD_AUTOTEST_OFF          =  0000000B     D
DSA$V_CRD_MENUTEST_OFF          =  00000010     D
DSA$V_CRD_MENUTEST_ON           =  00000012     D
DSA$V_CRD_TRACE                 =  00000011     D
DSA$V_USER                      =  0000001C     D
DSQA$K_DISPAT_AFTER_INIT        =  00000014     D
DSQA$K_DISPAT_BEFORE_INIT       =  00000013     D
DSQA$K_DISPAT_CALLTEST          =  00000015     D
DSQA$K_DISPAT_DONE_TESTS        =  00000018     D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =  00000001     D
DSQA$K_DISPAT_DSX$ENDPASS_END   =  00000002     D
DSQA$K_DISPAT_DSX$ENDPASS_MID   =  00000000     D
DSQA$K_DISPAT_DS_CLEANUP_BGN    =  00000003     D
DSQA$K_DISPAT_DS_CLEANUP_END    =  00000004     D
DSQA$K_DUMMY_T                  =  00000007     D
DSQA$K_ERROR_ERROR_BGN          =  00000006     D
DSQA$K_ERROR_ERROR_END          =  00000005     D
DSQA$K_KERNEL_INIT_CONTEXT      =  00000008     D
DSQA$K_LOOP_DSX$BGNSUB          =  00000009     D
DSQA$K_LOOP_DSX$CKLOOP          =  0000000B     D
DSQA$K_LOOP_DSX$ENDSUB          =  0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS     =  0000000E     D
DSQA$K_PARAM_DSX$GETDATA        =  0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL     =  0000000F     D
DSQA$K_PARAM_DSX$GETSTRING      =  00000010     D
DSQA$K_PARAM_DSX$GETVIELD       =  0000000D     D
DSQA$K_QA_DSX$BRANCH            =  00000011     D
DSQA$K_START_BEFORE_HEADER      =  00000017     D
DSQA$K_START_VRRESTART          =  00000012     D
DSQA$K_START_VRSTART            =  00000016     D
DSR$CHECK_AUTOTEST_OFF_SET         00000B04 RG  D   05
DSR$CHECK_MENUTEST_OFF_SET         00000AE0 RG  D   05
DSR$LOAD_CRD                       ********   X     05
DSR$RESTORE_CRD_VECTORS            ********   X     05
DSR$SETIMR_INI                     ********   X     05
DSR$UNLOAD_CRD                     ********   X     05
DSV$EXIT                           00000A37 RG  D   05
DSV$SETLOAD                        ********   X     05
DSX$CNTRLC                         00000A0C RG  D   05
DSX$GETTERM                        00000B23 RG  D   05
DSX$PRINT                          ********   X     05
DSX$PRINTI                         ********   X     05
DSX$WAITUS_USOVR                   ********   X     05
DS_CLEANUP                         ********   X     05
DS_ERRSUP                          ********   X     05
ESTKPTR                         =  00003800 RG  D   04
EXE$GB_CPUTYPE                     ********   X     05
EXE$GQ_SYSTIME                     ********   X     05
EXIT$DESBLK                        00000134 R   D   03
EXIT$HANDLR                        00000A64 R   D   05
FAB$C_BID                       =  00000003     D
FAB$C_BLN                       =  00000050     D
FAB$C_SEQ                       =  00000000     D
FAB$C_VAR                       =  00000002     D
FAB$L_ALQ                       =  00000010     D
FAB$L_FOP                       =  00000004     D
FAB$V_CHAN_MODE                 =  00000002     D
FAB$V_CIF                       =  00000019     D
FAB$V_FILE_MODE                 =  00000004     D
FAB$V_GET                       =  00000001     D
FAB$V_LNM_MODE                  =  00000000     D
FAB$V_PUT                       =  00000000     D
FAB$W_GBC                       =  00000048     D
FALSE                           =  00000000     D
FCB$SIZE                        =  00000030     D
GT_MENU_ERROR                      ********   X     05
HP$A_DEPENDENT                     00000032     D
HP$A_DEVICE                        00000018     D
HP$A_DVA                           0000001C     D
HP$A_LINK                          00000020     D
HP$B_DRIVE                         0000000B     D
HP$B_FLAGS                         0000000A     D
HP$Q_DEVICE                        00000000     D
HP$T_DEVICE                        0000000C     D
```

```
HP$T_TYPE                 00000026     D
HP$V_ALLOC              = 00000000     D
HP$V_WASALL             = 00000001     D
HP$W_SIZE                 00000008     D
HP$W_VECTOR               00000024     D
IDB$SIZE                = 00000020     D
IMAGE_HEADER              0000019C RG  D  03
IMAGE_HEADER_END          0000039C RG  D  03
INI$BRK                   00000200 RG  D  05
INI$LOAD_DEVICE           ********     X  05
INI$PTABLE                ********     X  05
INIT_COMMON               00000852 R   D  05
INIT_CONTEXT              0000073B RG  D  05
INIT_USERMODE             00000845 R   D  05
IO$M_CTRLCAST             ********     X  05
IO$M_NOW                  ********     X  05
IO$_READVBLK              ********     X  05
IO$_SETMODE               ********     X  05
IOC$K_DIAGPID           = 00660000   G  D
IRP$SIZE                = 00000050     D
ISTKPTR                 = 00003200 RG  D  04
JIB$C_LENGTH            = 0000006C     D
JIB$L_ARB               = 00000000     D
KB_CHECK                  00000936 RG  D  05
KB_CHECK_APT              0000093C RG  D  05
KB_CHECK_X                00000A0B R   D  05
KB_POLL                   ********     X  05
KSTKPTR                 = 00002A00 RG  D  04
LF                      = 0000000A     D
MAPMEM                    ********     X  05
MASK                    = 0000000F     D
MEMPOOL$INIT              ********     X  05
OFF                     = 00000000     D
ON                      = 00000001     D
POPT_BASE               = 00004400 RG  D  04
PCB$B_ASTACT              0000000C     D
PCB$B_ASTEN               0000000D     D
PCB$B_PRI                 0000000B     D
PCB$B_PRIB                0000002F     D
PCB$B_TYPE                0000000A     D
PCB$B_WEFC                0000002E     D
PCB$C_LENGTH              0000008C     D
PCB$K_LENGTH              0000008C     D
PCB$L_ARB                 00000084     D
PCB$L_ASTQBL              00000014     D
PCB$L_ASTQFL              00000010     D
PCB$L_EFC2P               00000058     D
PCB$L_EFC3P               0000005C     D
PCB$L_EFCS                00000050     D
PCB$L_EFCU                00000054     D
PCB$L_EFWM                0000004C     D
PCB$L_JIB                 00000078     D
PCB$L_OWNER               0000001C     D
PCB$L_PHD                 00000064     D
PCB$L_PHYPCB              00000018     D
PCB$L_PID                 00000060     D
PCB$L_PQB                 0000004C     D
PCB$L_SQBL                00000004     D
PCB$L_SQFL                00000000     D
PCB$L_STS                 00000024     D
PCB$L_UIC                 00000088     D
PCB$L_WSSWP               00000020     D
PCB$L_WTIME               00000028     D
PCB$Q_PRIV                0000007C     D
PCB$T_LNAME               00000068     D
PCB$T_TERMINAL            00000044     D
PCB$W_APTCNT              00000030     D
PCB$W_ASTCNT              00000038     D
PCB$W_BIGCNT              0000003A     D
PCB$W_BIOLM               0000003C     D
PCB$W_DIOCNT              0000003E     D
PCB$W_DIOLM               00000040     D
PCB$W_GPGCNT              00000034     D
PCB$W_GRP                 0000008A     D
PCB$W_MEM                 00000088     D
PCB$W_MTXCNT              0000000E     D
PCB$W_PPGCNT              00000036     D
PCB$W_PRCCNT              00000042     D
PCB$W_SIZE                00000008     D
PCB$W_STATE               0000002C     D
PCB$W_TMBU                00000032     D
PCB_BASE                = 00000078 RG  D  04
PHD$B_ASTLVL            = 000000CF     D
PHD$L_ESP               = 0000007C     D
PHD$L_KSP               = 00000078     D
PHD$L_P0BR              = 000000C8     D
PHD$L_P0LR              = 000000CC     D
PHD$L_P1BR              = 000000D0     D
PHD$L_P1LR              = 000000D4     D
PHD$L_PC                = 000000C0     D
PHD$L_PCB               = 00000078     D
PHD$L_PSL               = 000000C4     D
PHD$L_R0                = 00000088     D
PHD$L_R12               = 000000B8     D
PHD$L_SSP               = 00000080     D
PHD$L_USP               = 00000084     D
PHD_BASE                = 00000000 RG  D  04
PR$_ASTLVL              = 00000013     D
PR$_ESP                 = 00000001     D
PR$_IPL                 = 00000012     D
PR$_ISP                 = 00000004     D
PR$_KSP                 = 00000000     D
PR$_MAPEN               = 00000038     D
PR$_P0BR                = 00000008     D
PR$_P0LR                = 00000009     D
PR$_P1BR                = 0000000A     D
PR$_P1LR                = 0000000B     D
PR$_PCBB                = 00000010     D
PR$_SCBB                = 00000011     D
PR$_SID                 = 0000003E     D
PR$_SSP                 = 00000002     D
PR$_USP                 = 00000003     D
PRGBUF                    00000168 R   D  02
PSL$M_IPL               = 001F0000     D
```

```
PSL$V_IS                =  0000001A     D
PSL$V_TBIT              =  00000004     D
PSL_SAVE                   00000841 R   D   05
QA$MAIN                    ********     X   05
QIO$INITIALIZE             ********     X   05
QIO$MINCORE             =  00005E3C   G  D
RAB$B_RAC               =  0000001E     D
RAB$C_BID               =  00000001     D
RAB$C_BLN               =  00000044     D
RAB$C_SEQ               =  00000000     D
RAB$L_CTX               =  00000018     D
RAB$L_ROP               =  00000004     D
RAB$V_CCO               =  0000001F     D
RAB$V_CVT               =  0000001A     D
RCTRLC                     00000897 RG  D   05
RMS$INIT                   ********     X   05
RPB$B_BOOTNDT              000000A1     D
RPB$B_CONFREG              00000090     D
RPB$B_DEVTYP               00000066     D
RPB$B_HDRPGCNT             000000A0     D
RPB$B_RODEVTYP             0000001C     D
RPB$B_SLAVE                00000067     D
RPB$B_WAIT                 00000100     D
RPB$C_LENGTH               00000104     D
RPB$K_LENGTH               00000104     D
RPB$L_ADPPHY               0000005C     D
RPB$L_ADPVIR               00000060     D
RPB$L_BASE                 00000000     D
RPB$L_BOOTR0               0000001C     D
RPB$L_BOOTR1               00000020     D
RPB$L_BOOTR2               00000024     D
RPB$L_BOOTR3               00000028     D
RPB$L_BOOTR4               0000002C     D
RPB$L_BOOTR5               00000030     D
RPB$L_BUGCHK               000000FC     D
RPB$L_CHKSUM               00000008     D
RPB$L_CSRPHY               00000054     D
RPB$L_CSRVIR               00000058     D
RPB$L_FILLBN               0000003C     D
RPB$L_FILSIZ               00000040     D
RPB$L_HALTCODE             00000018     D
RPB$L_HALTPC               00000010     D
RPB$L_HALTPSL              00000014     D
RPB$L_IOVEC                00000034     D
RPB$L_IOVECSZ              00000038     D
RPB$L_ISP                  000000A4     D
RPB$L_MEMDSC               000000BC     D
RPB$L_PCBB                 000000A8     D
RPB$L_PFNCNT               0000004C     D
RPB$L_RESTART              00000004     D
RPB$L_RSTRTFLG             0000000C     D
RPB$L_SBR                  000000AC     D
RPB$L_SCBB                 000000B0     D
RPB$L_SISR                 000000B4     D
RPB$L_SLR                  000000B8     D
RPB$L_SVASPT               00000050     D
RPB$Q_PFNMAP               00000044     D
RPB$T_FILE                 00000068     D
RPB$V_AUTOTEST          =  0000000F     D
RPB$V_DIAG              =  00000004     D
RPB$V_INIBPT            =  00000002     D
RPB$V_MENUTEST          =  00000000     D
RPB$W_ROUBVEC              0000001E     D
RPB$W_UNIT                 00000064     D
SAVEFP                     00000148 R   D   03
SCB$L_ACCESS               00000020     D
SCB$L_ARITH                00000034     D
SCB$L_BREAK                0000002C     D
SCB$L_CHME                 00000044     D
SCB$L_CHMK                 00000040     D
SCB$L_CHMS                 00000048     D
SCB$L_CHMU                 0000004C     D
SCB$L_COMPAT               00000030     D
SCB$L_KNLSTK               00000008     D
SCB$L_MACHCHK              00000004     D
SCB$L_OPCCUS               00000014     D
SCB$L_OPCDEC               00000010     D
SCB$L_POWER                0000000C     D
SCB$L_RADRMOD              0000001C     D
SCB$L_ROPRAND              00000018     D
SCB$L_RXDB                 000000F8     D
SCB$L_SFTLVL1              00000084     D
SCB$L_SFTLVL10             000000A8     D
SCB$L_SFTLVL11             000000AC     D
SCB$L_SFTLVL12             000000B0     D
SCB$L_SFTLVL13             000000B4     D
SCB$L_SFTLVL14             000000B8     D
SCB$L_SFTLVL15             000000BC     D
SCB$L_SFTLVL2              00000088     D
SCB$L_SFTLVL3              0000008C     D
SCB$L_SFTLVL4              00000090     D
SCB$L_SFTLVL5              00000094     D
SCB$L_SFTLVL6              00000098     D
SCB$L_SFTLVL7              0000009C     D
SCB$L_SFTLVL8              000000A0     D
SCB$L_SFTLVL9              000000A4     D
SCB$L_TBIT                 00000028     D
SCB$L_TIMER                000000C0     D
SCB$L_TRANSL               00000024     D
SCB$L_TXDB                 000000FC     D
SCB$L_ZERO                 00000000     D
SCB_BASE                =  00000400  RG  D   04
SCB_IMAGE                  ********     X   05
SCB_UNKINT              =  00000C00  RG  D   04
SCH$GL_CURPCB              000007A0  RG  D   03
SCRIPT$INIT                ********     X   05
SCRIPT$STOP                ********     X   05
SGN$C_IRPCNT            =  00000040   G  D
SGN$C_NPAGEDYN          =  0000263C   G  D
SIZ...                  =  00000001     D
SIZE_TERM                  ********     X   05
SS$_NORMAL              =  00000001     D
SS$_NOTRAN                 ********     X   05
SSPROT                     0000013B R   D   02
```

```
SSTKPTR                         = 00003E00 RG D  04
STACK_BASE                      = 00001400 RG D  04
SUPBUF                            00000160 R  D  02
SWI$GL_FQBL                       000007A8 RG D  03
SWI$GL_FQFL                       000007A4 RG D  03
SYI$_SID                        = 00001001    D
SYS$ASSIGN                        ******** X    05
SYS$BINTIM                        ******** GX   05
SYS$CLREF                         ******** GX   05
SYS$CONNECT                       ******** GX   05
SYS$CREATE                        ******** GX   05
SYS$CRELOG                        ******** X    05
SYS$CRETVA                      = 800000C8 G D
SYS$DALLOC                        ******** GX   05
SYS$DCLEXH                        ******** GX   05
SYS$DELTVA                      = 80000110 G D
SYS$DISK                          ******** X    05
SYS$DXLEXH                      = 800000F0    D
SYS$EXIT                          ******** GX   05
SYS$GETCHN                        ******** GX   05
SYS$GETSYI                        ******** GX   05
SYS$INPUT                         0000014B R  D  02
SYS$OPEN                          ******** GX   05
SYS$OUTPUT                        00000155 R  D  02
SYS$QIOW                          ******** GX   05
SYS$SETAST                        ******** GX   05
SYS$SETDDIR                       ******** X    05
SYS$SETEXV                      = 80000208 G D
SYS$SETPRT                      = 80000230    D
SYS$SYSROOT                       ******** X    05
SYS$TRNLOG                        ******** X    05
SYSVEC                            00000133 R  D  02
TM_SIZ                          = 00001000    D
TRAN_ASSIGN                       00000594 R  D  05
TRUE                            = 00000001    D
T_BOTH_CRD_BITS_SET               00000030 R  D  02
T_WRONGCPU                        00000000 R  D  02
UCB$SIZE                        = 00000098    D
USTKPTR                         = 00004400 RG D  04
VCB$SIZE                        = 00000050    D
VRSETWIDTH                        ******** X    05
V_ENV                           = 00000001    D
V_LVL                           = 00000000    D
WCB$SIZE                        = 00000030    D
XDELBPT                           ********W GX   00
XDELTBIT                          ********W GX   00
XF_SIZ                          = 00000600    D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000170 (368.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
WORK	00000AEA (2794.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
_LAST	00000000 (0.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
CODE	00000B31 (2865.)	05 (5.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00004400-R	800 (5)	742 (5) 743 (5) 744 (5) 750 (5) 751 (5) 757 (5) 763 (5) 769 (5) 770 (5) 771 (5) 772 (5) 785 (5) 786 (5) 788 (5) 789 (5) 791 (5) 792 (5) 794 (5) 795 (5) 797 (5) 798 (5) 800 (5) 801 (5) 803 (5)
$$.TAB	=000000E4-R	593 (1)	584 (1) 587 (1) 590 (1) 593 (1)
$$.TABEND	=00000128-R	593 (1)	584 (1) 587 (1) 590 (1) 593 (1)
$$.TMP	=80000000	593 (1)	584 (1) 587 (1) 590 (1) 593 (1)
$$.TMP1	=00000001	1089 (5)	1083 (5) 1085 (5) 1087 (5) 1089 (5)
$$.TMP2	=000000CF	1089 (5)	1083 (5) 1085 (5) 1087 (5) 1089 (5)
$$ARGS	=00000003	719 (1)	719 (1)
$$N	=00000001	1465 (6)	#-1371 (5) #-1430 (6) #-1443 (6) 1448 (6) #-1465 (6) 938 (5)
$$T1	=00000001	1647 (6)	1039 (5) 1116 (5) 1233 (5) 1644 (6) 1647 (6) 719 (1)
$ER	=00000002	1639 (6)	#-1639 (6)
$MODULE	0000012C-R	715 (1)	1639 (6)
A	000007A4-R	643 (1)	643 (1) 646 (1)
APT	00000097-R	887 (5)	
ARB$C_LENGTH	=00000078		757 (5)
AX_BUFF	00000000-R	736 (5)	903 (5)
BEGIN	0000071E-R	1467 (6)	#-1464 (6) #-1477 (6) #-1591 (6) #-1901 (6)
BEGIN0	000005DC-R	1340 (5)	#-1247 (5) #-999 (5)
BEGIN_BLISS	0000071E-R	1468 (6)	
BIT...	=00000003	494 (1)	470 (1) 488 (1) 489 (1) 492 (1) 493 (1) 494 (1)
BOOT	00000000-R	847 (5)	
BOOT$L_ADP	000007D4-R	653 (1)	#-857 (5) #-862 (5)
BOOT$L_CSR	000007D8-R	654 (1)	#-863 (5)
BOOT$L_DEVTYP	000007E8-R	658 (1)	#-868 (5)
BOOT$L_R2	000007E0-R	656 (1)	#-865 (5)
BOOT$L_R5	000007E4-R	657 (1)	2034 (6) 2042 (6) 2109 (8) 2117 (8) #-866 (5) 916 (5)
BOOT$L_UNIT	000007DC-R	655 (1)	#-864 (5)
CCB$C_LENGTH	00000010		628 (1)
CCB$SIZE	=00000010	521 (1)	532 (1)
CLEAN_TRANSLATION	0000055C-R	1256 (5)	#-1144 (5) #-1193 (5)
CLI$K_BUFSIZ	=00000100	487 (1)	487 (1)
CLI$K_SIZE	00000444	487 (1)	
CLI$L_ADDRESS	00000018	487 (1)	
CLI$L_COMMAND	00000004	487 (1)	
CLI$L_DATA	0000001C	487 (1)	#-1435 (6)
CLI$L_FLAGS	00000000	487 (1)	
CLI$L_LAST	00000024	487 (1)	
CLI$L_NEXT	00000030	487 (1)	
CLI$L_PASS	0000002C	487 (1)	
CLI$L_SUBT	00000028	487 (1)	
CLI$L_TEST	00000020	487 (1)	
CLI$Q_BUFQWD	00000034	487 (1)	

```
CLI$Q_FILE                      000C0008     487     (1)     #-880     (5)
CLI$Q_SECTION                   00000010     487     (1)
CLI$Q_TIME                      0000043C     487     (1)
CLI$T_BUFFER                    0000003C     487     (1)
CLI$V_ADAPTER                  =00000018     487     (1)
CLI$V_ADR                      =0000000B     487     (1)
CLI$V_ASCII                    =00000013     487     (1)
CLI$V_BREAK                    =0000000A     487     (1)
CLI$V_BRIEF                    =0000001B     487     (1)
CLI$V_BYTE                     =0000000D     487     (1)
CLI$V_CLEAR                    =00000002     487     (1)
CLI$V_DEC                      =00000010     487     (1)
CLI$V_DEFAULT                  =0000000C     487     (1)
CLI$V_DEPOSIT                  =00000019     487     (1)
CLI$V_EVENT                    =00000008     487     (1)
CLI$V_EXAM                     =00000005     487     (1)
CLI$V_FLAGS                    =00000009     487     (1)
CLI$V_HEX                      =00000012     487     (1)
CLI$V_KERNEL                   =00000017     487     (1)
CLI$V_LOAD                     =00000006     487     (1)
CLI$V_LONG                     =0000000F     487     (1)
CLI$V_NOTNUF                   =00000001     487     (1)
CLI$V_OCT                      =00000011     487     (1)
CLI$V_PREG                     =0000001A     487     (1)
CLI$V_QA                       =00000007     487     (1)
CLI$V_QACKLOOPLOOPS            =0000001C     487     (1)
CLI$V_QAERRORPRINTS            =0000001B     487     (1)
CLI$V_QAMULTIPLEPASS           =0000001F     487     (1)
CLI$V_QASUBTESTLOOPS           =0000001E     487     (1)
CLI$V_QATESTLOOPS              =0000001D     487     (1)
CLI$V_REG                      =00000014     487     (1)
CLI$V_REQUIRED                 =00000000     487     (1)
CLI$V_RUN                      =00000015     487     (1)
CLI$V_SET                      =00000003     487     (1)
CLI$V_SHOW                     =00000004     487     (1)
CLI$V_VALSEC                   =00000016     487     (1)
CLI$V_WORD                     =0000000E     487     (1)
CLK$RE_INIT                     00000000-XR                    978     (5)
CMK$REFRESH                     00000829-R   1569     (6)     #-1539    (6)
CMK$_                          =00000004     488     (1)
CMK$_ASTEXIT                   =00000000     488     (1)
CMK$_CANTIM                    =0000000B     488     (1)
CMK$_HIBER                     =00000007     488     (1)
CMK$_INITSCB                   =00000008     488     (1)
CMK$_LAST                      =0000000E     488     (1)
CMK$_MMENABLE                  =00000003     488     (1)
CMK$_RCONSOLE                  =00000001     488     (1)
CMK$_REFRESH                   =00000005     488     (1)     #-1539    (6)
CMK$_SCHDWK                    =00000006     488     (1)
CMK$_SETIMR                    =0000000C     488     (1)
CMK$_SETIPL                    =00000009     488     (1)
CMK$_SETPRT                    =0000000D     488     (1)
CMK$_SGIPR                     =0000000A     488     (1)
CMK$_TCONSOLE                  =00000002     488     (1)
COND_DEBUG                      00000000-XR                   1039     (5)
COND_HANDLR                     00000000-XR                   1040     (5)       1595    (6)
CR                             =0000000D     511     (1)
```

```
CRB$SIZE                        =0000002C      518     (1)       529     (1)
CRD$K_CRD_INITIALIZATION        =00000000      494     (1)     #-1413    (6)
CRD$K_DS_CONTROL_C              =00000001      494     (1)
CRD$K_DS_FLAGS                  =00000000      494     (1)
CRD$K_FAILURE                   =00000000      494     (1)
CRD$K_HOOK_POINT                =00000000      494     (1)     #-1414    (6)
CRD$K_LAST_ACCESS_CODE          =00000002      494     (1)
CRD$K_LAST_DS_VARIABLE          =00000002      494     (1)
CRD$K_LAST_FUNCTION             =00000002      494     (1)
CRD$K_LAST_HOOK_POINT           =00000001      494     (1)
CRD$K_READ                      =00000000      494     (1)
CRD$K_SUCCESS                   =00000001      494     (1)
CRD$K_TYPECODE                  =00000001      494     (1)
CRD$K_WRITE                     =00000001      494     (1)
CRETVA$_ACMODE                  =0000000C      719     (1)
CRETVA$_INADR                   =00000004      719     (1)
CRETVA$_NARGS                   =00000003      719     (1)
CRETVA$_RETADR                  =00000008      719     (1)
CTL$GL_CCB                       0000039C-R    627     (1)
CTL$GL_CCBBASE                   0000079C-R    630     (1)       631     (1)
DB_SIZ                          =00000800      536     (1)       543     (1)
DDB$SIZE                        =00000018      517     (1)       528     (1)
DEF$Q_DEV                        00000000-XR                      1145    (5)       1209    (5)
DEF$Q_DIR                        00000000-XR                      1175    (5)       1218    (5)
DEV$V_FOD                       =0000000E                       #-1082    (5)
DEV$V_MBX                       =00000014                       #-1640    (6)
DIB$K_LENGTH                    =00000074                       #-1632    (6)     #-1634    (6)
DIB$L_DEVCHAR                   =00000000                        1641     (6)
DIB$L_DEVDEPEND                 =00000008                       #-1642    (6)
DL_SIZ                          =00000800      537     (1)       543     (1)
DM_SIZ                          =00000A00      538     (1)       543     (1)
DR_SIZ                          =00000800      539     (1)       543     (1)
DS$AQ_SSEND                      00000000-XR                     #-1067    (5)       717     (1)
DS$AQ_SYSSRV                     00000000-XR                      1045     (5)       1053    (5)       717     (1)
DS$AT_DDEV                       00000174-R    616     (1)       1132     (5)       1957    (6)
DS$AT_DDIR                       0000014C-R    614     (1)       1214     (5)       1217    (5)       1960    (6)
DS$AX_ARB                       =0000026C-R    751     (5)       1532     (6)
DS$AX_JIB                       =00000200-R    750     (5)       1530     (6)
DS$AX_SOFTPCB                   =000002E4-R    757     (5)       1522     (6)     #-1694    (6)       635     (1)
DS$A_PRGBGN                     =00000200      508     (1)       509     (1)       731     (5)
DS$CLI                           00000000-XR                      1473     (6)       1779    (6)
DS$CNTRLC                        00000000-XR                      1456     (6)       925     (5)
DS$FAB_INPUT                     00000000-R    583     (1)       1083     (5)       587     (1)
DS$FAB_OUTPUT                    00000094-R    589     (1)       1087     (5)       593     (1)
DS$GA_BUFPTR                     0000084C-R    702     (1)
DS$GA_CHKLPPC                    0000082C-R    693     (1)
DS$GA_CRD_DS_INTERFACE           00000000-XR                      1416     (6)
DS$GA_DS_CTRL_C_FIRST            00000128-R    596     (1)     #-1736    (6)     #-1738    (6)     #-1740    (6)     #-927     (5)
DS$GA_DS_CTRL_C_SECOND           0000012C-R    599     (1)     #-1757    (6)     #-1759    (6)     #-1761    (6)     #-928     (5)
DS$GA_LASTADR                    000007F4-R    663     (1)
DS$GA_LOOPADR                    00000830-R    694     (1)
DS$GA_PTABLE                     00000000-XR                      1942     (6)
DS$GA_SELECTS                    00000000-XR                      1943     (6)
DS$GB_BYTEBUF                    00000850-R    703     (1)
DS$GB_INHIBIT_NAMING             00000852-R    705     (1)     #-1462    (6)
DS$GB_MM_ENB                     00000000-XR                     #-961     (5)
DS$GB_TYPECODE                   00000851-R    704     (1)     #-1586    (6)
```

```
DS$GK_SID                       00000000-XR                    #-1236  (5)    #-934   (5)
DS$GL_BUFCNT                    00000000-XR                    #-958   (5)    #-959   (5)
DS$GL_BUFLEN                    00000848-R   701   (1)
DS$GL_CLIBASE                   00000000-XR                     1434   (6)     879    (5)
DS$GL_DEVERR_COUNT              00000810-R   683   (1)
DS$GL_EFC0                      000007EC-R   661   (1)
DS$GL_EFC1                      000007F0-R   662   (1)
DS$GL_ERRCNT                    00000800-R   676   (1)
DS$GL_ERRSUP_COUNT              00000818-R   687   (1)
DS$GL_FLAGS                     000007F8-R   673   (1)  #-1036  (5)    1092   (5)  #-1648  (6)    1701   (6)
                                                        1705   (6)  #-1742  (6)  #-1752  (6)  #-1763  (6)
                                                        1848   (6)    1877   (6)    1939   (6)  #-896   (5)
DS$GL_FSTTEST                   00000820-R   690   (1)
DS$GL_HARDERR_COUNT             00000804-R   677   (1)
DS$GL_LSTTEST                   00000824-R   691   (1)
DS$GL_NUMTEST                   0000081C-R   689   (1)
DS$GL_PCBVEC                    000007A0-R   634   (1)
DS$GL_PREPERR_COUNT             00000808-R   679   (1)
DS$GL_RADIX                     00000840-R   698   (1)  #-1245  (5)  #-897   (5)
DS$GL_RUNTIM                    00000834-R   695   (1)
DS$GL_SIZE                      00000844-R   699   (1)
DS$GL_SOFTERR_COUNT             0000080C-R   681   (1)
DS$GL_SUBTEST                   00000828-R   692   (1)
DS$GL_SYSERR_COUNT              00000814-R   685   (1)
DS$GL_TERMCHAR                  00000AE2-R   710   (1)  #-1113  (5)  #-2174  (8)
DS$GL_TIMESEC                   00000838-R   696   (1)
DS$GL_WAITIM                    0000083C-R   697   (1)
DS$GQ_BUFQWD                    00000848-R   700   (1)
DS$GQ_CURYEAR                   00000000-XR                    #-973   (5)  #-975   (5)    976    (5)  #-977   (5)
DS$GQ_DAYTIM                    00000000-XR                    #-971   (5)
DS$GQ_EFL                       000007EC-R   660   (1)
DS$GT_ASCIBUF                   00000853-R   707   (1)
DS$GT_BUFFER                    00000862-R   708   (1)   1644   (6)
DS$GT_LEGAL                     00000060-R   573   (1)   1443   (6)
DS$GT_NEWYEAR                   00000000-XR                    #-972   (5)    974    (5)
DS$GT_PROMPT                    00000000-XR                     1465   (6)
DS$GT_START                     00000000-XR                     1448   (6)
DS$GT_STRBUF                    00000A62-R   709   (1)
DS$GW_TTIN                      000007FC-R   674   (1)  #-1074  (5)  #-1116  (5)  #-1636  (6)  #-1644  (6)
                                                        #-1647  (6)
DS$GW_TTOUT                     000007FE-R   675   (1)  #-1104  (5)
DS$INITSCB                      00000000-XR                     949   (5)
DS$K_ASCIBUF                    =0000000F    669   (1)     707   (1)
DS$K_BUFSIZ                     =00000200    670   (1)     708   (1)
DS$K_CCB                        =00000040    625   (1)     628   (1)
DS$K_ERROR                      =00000002    470   (1)
DS$K_NORMAL                     =00000001    470   (1)
DS$K_NYSIZ                      00000000-XR                    #-973   (5)
DS$K_PRGSIZ                     =0000F800    509   (1)     731   (5)
DS$K_PRINTB                     =00000002    493   (1)
DS$K_PRINTF                     =00000001    493   (1)  #-1371  (5)  #-1430  (6)  #-1443  (6)  #-1448  (6)
DS$K_PRINTI                     =00000000    493   (1)
DS$K_PRINTX                     =00000003    493   (1)
DS$K_SEVERE                     =00000004    470   (1)
DS$K_SSSIZE                     00000000-XR                    #-1044  (5)
DS$K_STRBUF                     =00000080    671   (1)     709   (1)
DS$K_SUBSYS                     =00000066    470   (1)     470   (1)    471    (1)
```

```
DS$K_TYPE_ABORT_PROGRAM         =00000014     493     (1)
DS$K_TYPE_ABORT_TEST            =00000013     493     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     493     (1)
DS$K_TYPE_COMMAND_OUT           =00000016     493     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     493     (1)
DS$K_TYPE_DS_PROMPT             =00000001     493     (1)
DS$K_TYPE_DS_START              =0000001D     493     (1)    #-1443   (6)    #-1448   (6)
DS$K_TYPE_ERRDEV                =00000008     493     (1)
DS$K_TYPE_ERRHARD               =00000006     493     (1)
DS$K_TYPE_ERROR_BODY            =00000009     493     (1)
DS$K_TYPE_ERROR_END             =0000000A     493     (1)
DS$K_TYPE_ERRPREP               =0000001B     493     (1)
DS$K_TYPE_ERRSOFT               =00000007     493     (1)
DS$K_TYPE_ERRSUP                =00000004     493     (1)
DS$K_TYPE_ERRSYS                =00000005     493     (1)
DS$K_TYPE_ERR_HALT              =0000000D     493     (1)
DS$K_TYPE_EXCEPTION             =0000000C     493     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     493     (1)
DS$K_TYPE_FIRST_PASS            =00000011     493     (1)
DS$K_TYPE_GENERAL               =00000000     493     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     493     (1)    #-1371   (5)    #-1430   (6)
DS$K_TYPE_NO_TESTS              =00000012     493     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     493     (1)
DS$K_TYPE_PROGRAM_END           =00000010     493     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     493     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     493     (1)
DS$K_TYPE_QIO_INVADP            =00000024     493     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     493     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     493     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     493     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     493     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     493     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     493     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     493     (1)
DS$K_TYPE_START_ERR             =00000018     493     (1)
DS$K_TYPE_START_LIST            =00000025     493     (1)
DS$K_TYPE_SUMMARY               =0000000E     493     (1)
DS$K_TYPF_USER_PROMPT           =00000002     493     (1)
DS$K_WARNING                    =00000000     470     (1)
DS$LOADPCS                       00000000-XR                  942     (5)
DS$L_USERCNTRLC                  00000130-R    602     (1)    #-1747   (6)    #-1749   (6)    #-1750   (6)    #-1844   (6)
                                                              #-929    (5)
DS$M_ABRTFLG                    =00000040     489     (1)
DS$M_BADTIME                    =00100000     489     (1)
DS$M_BATCH                      =00400000     489     (1)
DS$M_BRKCLR                     =00001000     489     (1)
DS$M_BRKPT                      =00000800     489     (1)
DS$M_CHARFLG                    =00000100     489     (1)
DS$M_CMDFLG                     =00000080     489     (1)    #-895    (5)
DS$M_CTRLC                      =00000001     489     (1)    #-1648   (6)    #-1741   (6)    #-1751   (6)    #-1762   (6)
DS$M_CTRLO                      =00010000     489     (1)    #-1741   (6)    #-1751   (6)    #-1762   (6)
DS$M_DEVFLG                     =00000200     489     (1)
DS$M_DISABLCC                   =01000000     489     (1)
DS$M_DONFLG                     =00002000     489     (1)
DS$M_ERRFLG                     =00000010     489     (1)
DS$M_EXCEPT                     =00080000     489     (1)
DS$M_EXETST                     =00040000     489     (1)
```

```
DS$M_HLTFLG                     =00000008      489     (1)
DS$M_LODFLG                     =00000002      489     (1)     #-895     (5)
DS$M_MEMMGT                     =00008000      489     (1)
DS$M_OUTPUT                     =00800000      489     (1)
DS$M_RUBFLG                     =00000020      489     (1)
DS$M_SCRIPT                     =00200000      489     (1)
DS$M_SETIMR                     =02000000      489     (1)
DS$M_STRFLG                     =00000004      489     (1)
DS$M_SUBT                       =00004000      489     (1)
DS$M_SYSFLG                     =00000400      489     (1)
DS$M_TIMRON                     =00020000      489     (1)
DS$RAB_INPUT                     00000050-R    586     (1)       1085    (5)
DS$RAB_OUTPUT                    000000E4-R    592     (1)       1089    (5)
DS$SOFTIPL5                      00000202-R    1004    (5)
DS$USERMODE                      0000020D-R    1035    (5)
DS$USERMODE_X                    0000055B-R    1249    (5)     #-1099    (5)
DS$V_ABRTFLG                    =00000006      489     (1)
DS$V_BADTIME                    =00000014      489     (1)
DS$V_BATCH                      =00000016      489     (1)     #-1091    (5)
DS$V_BRKCLR                     =0000000C      489     (1)
DS$V_BRKPT                      =0000000B      489     (1)
DS$V_CHARFLG                    =00000008      489     (1)
DS$V_CMDFLG                     =00000007      489     (1)
DS$V_CTRLC                      =00000000      489     (1)     #-1700    (6)
DS$V_CTRLO                      =00000010      489     (1)
DS$V_DEVFLG                     =00000009      489     (1)
DS$V_DISABLCC                   =00000018      489     (1)     #-1704    (6)     #-1849    (6)     #-1850    (6)
DS$V_DONFLG                     =0000000D      489     (1)
DS$V_ERRFLG                     =00000004      489     (1)
DS$V_EXCEPT                     =00000013      489     (1)
DS$V_EXETST                     =00000012      489     (1)
DS$V_HLTFLG                     =00000003      489     (1)
DS$V_LODFLG                     =00000001      489     (1)     #-1877    (6)     #-1939    (6)
DS$V_MEMMGT                     =0000000F      489     (1)
DS$V_OUTPUT                     =00000017      489     (1)
DS$V_RUBFLG                     =00000005      489     (1)
DS$V_SCRIPT                     =00000015      489     (1)
DS$V_SETIMR                     =00000019      489     (1)
DS$V_STRFLG                     =00000002      489     (1)
DS$V_SUBT                       =0000000E      489     (1)
DS$V_SYSFLG                     =0000000A      489     (1)
DS$V_TIMRON                     =00000011      489     (1)
DS$_ARITH                       =006600D0      470     (1)
DS$_ASBE                        =00660118      470     (1)
DS$_BADLINK                     =006600F0      470     (1)
DS$_BADTYPE                     =006600E8      470     (1)
DS$_BIIC                        =00660120      470     (1)
DS$_CHME                        =006600A8      470     (1)
DS$_CHMK                        =006600E0      470     (1)
DS$_DEVNAME                     =00660108      470     (1)
DS$_ERROR                       =00660002      470     (1)
DS$_FHWE                        =00660068      470     (1)
DS$_FRAGBUF                     =00660080      470     (1)
DS$_ICBUSY                      =006600C8      470     (1)
DS$_ICERR                       =006600C0      470     (1)
DS$_IHWE                        =00660060      470     (1)
DS$_ILLCHAR                     =00660018      470     (1)
```

```
DS$_ILLPAGCNT                   =00660078      470     (1)
DS$_ILLUNIT                     =00660100      470     (1)
DS$_INSFMEM                     =00660050      470     (1)
DS$_IPL2HI                      =006600B8      470     (1)
DS$_IVADDR                      =00660040      470     (1)
DS$_IVVECT                      =00660038      470     (1)
DS$_KRNLSTK                     =00660090      470     (1)
DS$_LOGIC                       =00660070      470     (1)
DS$_MCHK                        =00660088      470     (1)
DS$_MMOFF                       =00660058      470     (1)
DS$_NEEDUNIT                    =006600F8      470     (1)
DS$_NODE                        =00660128      470     (1)
DS$_NOPCS                       =00660110      470     (1)
DS$_NORMAL                      =00660001      470     (1)
DS$_NOSUPPORT                   =006600B1      470     (1)
DS$_NOTDON                      =00660030      470     (1)
DS$_NOTIMP                      =006600B0      470     (1)       470     (1)
DS$_NULLSTR                     =00660010      470     (1)
DS$_OVERFLOW                    =00660008      470     (1)
DS$_POWER                       =00660098      470     (1)
DS$_PROGERR                     =00660020      470     (1)
DS$_SEVERE                      =00660004      470     (1)
DS$_TRANSL                      =006600A0      470     (1)
DS$_TRUNCATE                    =00660028      470     (1)
DS$_UNEXPINT                    =006600D8      470     (1)
DS$_VASFULL                     =00660048      470     (1)
DS$_WARNING                     =00660000      470     (1)       470     (1)
DSA$AL_APTMAIL                   0000FE00                         728     (3)       852     (5)
DSA$GL_EVENT                     0000FE48                       #-1690    (6)
DSA$GL_FLAGS                     0000FE00                       #-1239    (5)      1345     (5)      1346     (5)      1347     (5)
                                                                  1348     (5)      1382     (6)      1395     (6)      1423     (6)
                                                                  1424     (6)      1425     (6)      1426     (6)    #-1439     (6)
                                                                  1460     (6)      1464     (6)      1517     (6)      1693     (6)
                                                                  1898     (6)      2023     (6)      2098     (8)    #-855      (5)
DSA$GL_SID                       0000FE14                         1229     (5)    #-1236     (5)    #-892      (5)    #-893      (5)
                                                                #-935      (5)    #-938      (5)    #-944      (5)      946      (5)
DSA$M_APT                       =80000000                       #-1439     (6)
DSA$M_CRD_AUTOTEST_OFF          =00000800                       #-1437     (6)
DSA$M_CRD_MENUTEST_OFF          =00010000                       #-1438     (6)
DSA$M_DFLTFLGS                  =00003000      506     (1)      #-1238     (5)    #-855      (5)
DSA$M_OPER                      =00001000                         506      (1)
DSA$M_PROMPT                    =00002000                         506      (1)
DSA$M_USER                      =10000000                       #-1238     (5)
DSA$V_APT                       =0000001F                       #-1459     (6)    #-1464     (6)
DSA$V_BINARY                    =00000001                       #-1423     (6)
DSA$V_CRD_AUTOTEST_OFF          =0000000B                       #-1347     (5)    #-1382     (6)    #-1426     (6)    #-2097     (8)
DSA$V_CRD_MENUTEST_OFF          =00000010                       #-1345     (5)    #-1395     (6)    #-1424     (6)    #-2022     (6)
DSA$V_CRD_MENUTEST_ON           =00000012                       #-1346     (5)    #-1425     (6)
DSA$V_CRD_TRACE                 =00000011                       #-1348     (5)
DSA$V_USER                      =0000001C                       #-1517     (6)    #-1692     (6)    #-1898     (6)
DSQA$K_DISPAT_AFTER_INIT        =00000014      492     (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013      492     (1)
DSQA$K_DISPAT_CALLTEST          =00000015      492     (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018      492     (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001      492     (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002      492     (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000      492     (1)
```

```
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003     492    (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004     492    (1)
DSQA$K_DUMMY_T                  =00000007     492    (1)
DSQA$K_ERROR_ERROR_BGN          =00000006     492    (1)
DSQA$K_ERROR_ERROR_END          =00000005     492    (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008     492    (1)    #-1516   (6)
DSQA$K_LOOP_DSX$BGNSUB          =00000009     492    (1)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B     492    (1)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A     492    (1)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E     492    (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C     492    (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F     492    (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010     492    (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D     492    (1)
DSQA$K_QA_DSX$BRANCH            =00000011     492    (1)
DSQA$K_START_BEFORE_HEADER      =00000017     492    (1)
DSQA$K_START_VRRESTART          =00000012     492    (1)
DSQA$K_START_VRSTART            =00000016     492    (1)
DSR$CHECK_AUTOTEST_OFF_SET       00000B04-R   2093   (8)    #-1358   (5)
DSR$CHECK_MENUTEST_OFF_SET       00000AE0-R   2018   (6)    #-1361   (5)    #-1385   (6)
DSR$LOAD_CRD                     00000000-XR                   1402   (6)
DSR$RESTORE_CRD_VECTORS          00000000-XR                   1341   (5)
DSR$SETIMR_INI                   00000000-XR                   979    (5)
DSR$UNLOAD_CRD                   00000000-XR                   1892   (6)
DSV$EXIT                         00000A37-R   1876   (6)      1373   (5)      1432   (6)
DSV$SETLOAD                      00000000-XR                   881    (5)
DSX$CNTRLC                       00000A0C-R   1842   (6)
DSX$GETTERM                      00000B23-R   2172   (8)
DSX$PRINT                        00000000-XR                   1371   (5)      1430   (6)      1443   (6)      1448   (6)
DSX$PRINTI                       00000000-XR                   1465   (6)      938    (5)
DSX$WAITUS_USOVR                 00000000-XR                 #-947    (5)
DS_CLEANUP                       00000000-XR                 #-1878   (6)    #-1940   (6)
DS_ERRSUP                        00000000-XR                   1639   (6)
ESTKPTR                         =00003800-R   795    (5)    #-1544   (6)
EXE$GB_CPUTYPE                   00000000-XR                 #-894    (5)
EXE$GQ_SYSTIME                   00000000-XR                 #-977    (5)
EXIT$DESBLK                      00000134-R   605    (1)      1043   (5)
EXIT$HANDLR                      00000A64-R   1937   (6)      607    (1)
FAB$C_BID                       =00000003                      584    (1)      590    (1)
FAB$C_BLN                       =00000050                      584    (1)      590    (1)
FAB$C_SEQ                       =00000000                      584    (1)      590    (1)
FAB$C_VAR                       =00000002                      584    (1)      590    (1)
FAB$L_ALQ                       =00000010                      584    (1)      590    (1)
FAB$L_FOP                       =00000004                      584    (1)      590    (1)
FAB$V_CHAN_MODE                 =00000002                      584    (1)      590    (1)
FAB$V_CIF                       =00000019                      590    (1)
FAB$V_FILE_MODE                 =00000004                      584    (1)      590    (1)
FAB$V_GET                       =00000001                      584    (1)
FAB$V_LNM_MODE                  =00000000                      584    (1)      590    (1)
FAB$V_PUT                       =00000000                      590    (1)
FAB$W_GBC                       =00000048                      584    (1)      590    (1)
FALSE                           =00000000     494    (1)
FCB$SIZE                        =00000030     524    (1)      535    (1)
GT_MENU_ERROR                    00000000-XR                   1430   (6)
HP$B_FLAGS                       0000000A                      1949   (6)      1951   (6)
HP$Q_DEVICE                      00000000                      1952   (6)
HP$V_ALLOC                      =00000000                    #-1948   (6)
```

```
HP$V_WASALL                     =00000001                       #-1950    (6)
IDB$SIZE                        =00000020       519     (1)       530     (1)
IMAGE_HEADER                     0000019C-R     618     (1)
IMAGE_HEADER_END                 0000039C-R     619     (1)
INI$BRK                          00000200-R     1001    (5)     #-1007    (5)     #-917     (5)
INI$LOAD_DEVICE                  00000000-XR                       988     (5)
INI$PTABLE                       00000000-XR                     #-1243    (5)       984     (5)
INIT_COMMON                      00000852-R     1584    (6)     #-1568    (6)
INIT_CONTEXT                     0000073B-R     1515    (6)     #-1469    (6)     #-1880    (6)       986     (5)
INIT_USERMODE                    00000845-R     1579    (6)     #-1519    (6)
IO$M_CTRLCAST                    00000000-XR                     #-1116    (5)     #-1647    (6)
IO$M_NOW                         00000000-XR                     #-1644    (6)
IO$_READVBLK                     00000000-XR                     #-1644    (6)
IO$_SETMODE                      00000000-XR                     #-1116    (5)     #-1647    (6)
IOC$K_DIAGPID                   =00660000       471     (1)     #-1523    (6)
IRP$SIZE                        =00000050       520     (1)       531     (1)
ISTKPTR                         =00003200-R     792     (5)     #-1540    (6)     #-849     (5)     #-890     (5)
JIB$C_LENGTH                    =0000006C                         751     (5)       757     (5)
JIB$L_ARB                       =00000000       481     (1)     #-1534    (6)
KB_CHECK                         00000936-R     1689    (6)
KB_CHECK_APT                     0000093C-R     1691    (6)
KB_CHECK_X                       00000A0B-R     1803    (6)     #-1696    (6)     #-1702    (6)     #-1707    (6)
KB_POLL                          00000000-XR                       1698    (6)
KSTKPTR                         =00002A00-R     789     (5)       1541    (6)
LF                              =0000000A       512     (1)
MAPMEM                           00000000-XR                       955     (5)
MASK                            =0000000F       549     (1)
MEMPOOL$INIT                     00000000-XR                      1241     (5)
OFF                             =00000000       494     (1)
ON                              =00000001       494     (1)
POPT_BASE                       =00004400-R     803     (5)     #-905     (5)
PCB$B_ASTACT                     0000000C                        #-1526    (6)     #-1694    (6)
PCB$L_ARB                        00000084                        #-1533    (6)
PCB$L_ASTQBL                     00000014                        #-1525    (6)
PCB$L_ASTQFL                     00000010                          1524    (6)       1525    (6)
PCB$L_JIB                        00000078                        #-1531    (6)
PCB$L_PID                        00000060                        #-1523    (6)
PCB_BASE                        =00000078-R     743     (5)     #-899     (5)
PHD$B_ASTLVL                    =000000CF                       #-1561    (6)
PHD$L_ESP                       =0000007C                       #-1545    (6)
PHD$L_KSP                       =00000078                       #-1543    (6)
PHD$L_P0BR                      =000000C8                       #-1558    (6)
PHD$L_POLRASTL                  =000000CC                       #-1559    (6)
PHD$L_P1BR                      =000000D0                       #-1562    (6)
PHD$L_P1LR                      =000000D4                       #-1563    (6)
PHD$L_PC                        =000000C0                       #-1537    (6)     #-1567    (6)
PHD$L_PCB                       =00000078                         743     (5)
PHD$L_PSL                       =000000C4                       #-1557    (6)
PHD$L_R0                        =00000088                       #-1564    (6)
PHD$L_R12                       =000000B8                         1553    (6)
PHD$L_SSP                       =00000080                       #-1547    (6)
PHD$L_USP                       =00000084                       #-1549    (6)
PHD_BASE                        =00000000-R     742     (5)       1536    (6)
PR$_ASTLVL                      =00000013                       #-1560    (6)
PR$_ESP                         =00000001                       #-1544    (6)     #-1545    (6)
PR$_IPL                         =00000012                       #-1565    (6)     #-1570    (6)     #-848     (5)     #-888     (5)
PR$_ISP                         =00000004                       #-1540    (6)
```

```
PR$_KSP                 =00000000                       #-1542   (6)     #-1543   (6)
PR$_MAPEN               =00000038                       #-960    (5)
PR$_P0BR                =00000008                       #-1558   (6)
PR$_P0LR                =00000009                       #-1559   (6)
PR$_P1BR                =0000000A                       #-1562   (6)
PR$_P1LR                =0000000B                       #-1563   (6)
PR$_PCBB                =00000010                       #-899    (5)
PR$_SCBB                =00000011                       #-898    (5)
PR$_SID                 =0000003E                       #-859    (5)     #-892    (5)
PR$_SSP                 =00000002                       #-1546   (6)     #-1547   (6)
PR$_USP                 =00000003                       #-1548   (6)     #-1549   (6)
PRGBUF                   00000168-R   731    (5)         1038    (5)       1120    (5)
PSL$M_IPL               =001F0000                       #-1571   (6)
PSL$V_IS                =0000001A    488    (1)         #-1539   (6)
PSL$V_TBIT              =00000004                       #-1572   (6)     #-1573   (6)
PSL_SAVE                 00000841-R   1577   (6)         #-1538   (6)       1572    (6)
QA$MAIN                  00000000-XR                       1516    (6)
QIO$INITIALIZE           00000000-XR                       957     (5)
QIO$MINCORE             =00005E3C    543    (1)
RAB$B_RAC               =0000001E                         587     (1)       593     (1)
RAB$C_BID               =00000001                         587     (1)       593     (1)
RAB$C_BLN               =00000044                         587     (1)       593     (1)
RAB$C_SEQ               =00000000                         587     (1)       593     (1)
RAB$L_CTX               =00000018                         587     (1)       593     (1)
RAB$L_ROP               =00000004                         587     (1)       593     (1)
RAB$V_CCO               =0000001F                         593     (1)
RAB$V_CVT               =0000001A                         587     (1)
RCTRLC                   00000897-R   1631   (6)         1116    (5)       1647    (6)
RMS$INIT                 00000000-XR                       998     (5)
RPB$B_DEVTYP             00000066                        #-867    (5)
RPB$L_ADPPHY             0000005C                        #-862    (5)
RPB$L_BOOTR2             00000024                        #-865    (5)
RPB$L_BOOTR5             00000030                        #-866    (5)
RPB$L_CSRPHY             00000054                        #-863    (5)
RPB$T_FILE               00000068                          869    (5)
RPB$V_AUTOTEST          =0000000F    498    (1)         #-2116   (8)
RPB$V_DIAG              =00000004                       #-2033   (6)     #-2108   (8)
RPB$V_INIBPT            =00000002                       #-916    (5)
RPB$V_MENUTEST          =00000000    502    (1)         #-2041   (6)
RPB$W_UNIT               00000064                        #-864    (5)
SAVEFP                   00000148-R   612    (1)         #-1246   (5)     #-1581   (6)
SCB$L_BREAK              0000002C                        #-915    (5)
SCB$L_TBIT               00000028                        #-914    (5)
SCB_BASE                =00000400-R   769    (5)
SCB_IMAGE                00000000-XR                     #-898    (5)       913     (5)
SCB_UNKINT              =00000C00-R   771    (5)
SCH$GL_CURPCB            000007A0-R   633    (1)
SCRIPT$INIT              00000000-XR                       1450    (6)
SCRIPT$STOP              00000000-XR                       1709    (6)
SGN$C_IRPCNT            =00000040    526    (1)           531     (1)
SGN$C_NPAGEDYN          =0000263C    535    (1)           543     (1)
SIZ...                  =00000001    489    (1)           489     (1)
SIZE_TERM                00000000-XR                       993     (5)
SS$_NORMAL              =00000001    469    (1)         #-2175   (8)
SS$_NOTRAN               00000000-XR                     #-1197   (5)
SSPROT                   0000013B-R   719    (1)          1041    (5)       1042    (5)
SSTKPTR                 =00003E00-R   798    (5)         #-1546   (6)
```

```
STACK_BASE                      =00001400-R  785  (5)
STJPBUF                          00000160-R  728  (3)      1037  (5)      1119  (5)
SWI$GL_FQBL                      000007A8-R  645  (1)
SWI$GL_FQFL                      000007A4-R  642  (1)
SYI$_SID                        =00001001                #-1230  (5)
SYS$ASSIGN                       00000000-XR              1305  (5)
SYS$BINTIM                       00000000-XR               976  (5)
SYS$CLREF                        00000000-XR              1453  (6)
SYS$CONNECT                      00000000-XR              1085  (5)      1089  (5)
SYS$CREATE                       00000000-XR              1087  (5)
SYS$CRELOG                       00000000-XR              1212  (5)      1964  (6)
SYS$CRETVA                      =8000C0C8    555  (1)      1037  (5)      1038  (5)      1042  (5)      1119  (5)
                                                          1120  (5)
SYS$DALLOC                       00000000-XR              1952  (6)
SYS$DCLEXH                       00000000-XR              1043  (5)
SYS$DELTVA                      =80000110    557  (1)      1041  (5)
SYS$DISK                         00000000-XR              1210  (5)
SYS$DXLEXH                      =800000F0    556  (1)
SYS$EXIT                         00000000-XR              1907  (6)
SYS$GETCHN                       00000000-XR              1079  (5)      1110  (5)      1636  (6)
SYS$GETSYI                       00000000-XR              1233  (5)
SYS$INPUT                        00000149-R   722  (2)      1069  (5)       584  (1)
SYS$OPEN                         00000000-XR              1083  (5)
SYS$OUTPUT                       00000155-R   725  (3)      1095  (5)       590  (1)
SYS$QIOW                         00000000-XR              1116  (5)      1644  (6)      1647  (6)
SYS$SETAST                       00000000-XR              1588  (6)
SYS$SETDDIR                      00000000-XR              1219  (5)      1971  (6)
SYS$SETEXV                      =80000208    558  (1)      1039  (5)
SYS$SETPRT                      =80000230    559  (1)
SYS$SYSROOT                      00000000-XR              1140  (5)      1962  (6)
SYS$TRNLOG                       00000000-XR              1141  (5)      1189  (5)      1293  (5)
SYSVEC                           00000133-R   717  (1)       719  (1)
TM_SIZ                          =00001000    540  (1)       543  (1)
TRAN_ASSIGN                      00000594-R  1281  (5)  #-1070  (5)  #-1096  (5)
TRUE                            =00000001    494  (1)
T_BOTH_CRD_BITS_SET              00000030-R   570  (1)      1371  (5)
T_WRONGCPU                       00000000-R   567  (1)       938  (5)
UCB$SIZE                        =00000098    516  (1)       527  (1)
USTKPTR                         =00004400-R  801  (5)  #-1548  (6)
VCB$SIZE                        =00000050    522  (1)       533  (1)
VRSETWIDTH                       00000000-XR              1436  (6)
V_ENV                           =00000001    490  (1)
V_LVL                           =00000000    490  (1)
WCB$SIZE                        =00000030    523  (1)       534  (1)
XDELBPT                          00000000-XR          #-1005  (5)       463  (1)       915  (5)
XDELTBIT                         00000000-XR              463  (1)  #-911  (5)
XF_SIZ                          =00000600    541  (1)       543  (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$$R_TABINIT	1	584 (1)	584 (1) 587 (1) 590 (1) 593 (1)
$$R_VBFSET	1	584 (1)	584 (1) 587 (1) 590 (1) 593 (1)
$ARBDEF	1	475 (1)	475 (1)
$BINTIM_S	1	976 (5)	976 (5)
$CCBDEF	1	474 (1)	474 (1)
$CLREF_S	1	1453 (6)	1453 (6)
$CONNECT	1	1085 (5)	1085 (5) 1089 (5)
$CRD_LITERALS	2	494 (1)	494 (1)
$CREATE	1	1087 (5)	1087 (5)
$CRETVA	1	719 (1)	719 (1)
$CRETVADEF	1	719 (1)	719 (1)
$CRETVA_G	1	1042 (5)	1042 (5)
$CRETVA_S	1	1037 (5)	1037 (5) 1038 (5) 1119 (5) 1120 (5)
$D1_PRINT_S	1	938 (5)	1371 (5) 1430 (6) 1443 (6) 1448 (6) 1465 (6) 938 (5)
$DALLOC_S	1	1952 (6)	1952 (6)
$DCLEXH_S	1	1043 (5)	1043 (5)
$DEF	1	494 (1)	
$DEFINI	1	472 (1)	472 (1) 473 (1) 474 (1) 475 (1) 476 (1) 477 (1) 478 (1) 479 (1) 480 (1) 482 (1) 483 (1) 484 (1) 485 (1) 486 (1) 491 (1) 584 (1) 587 (1)
$DELTVA_G	1	1041 (5)	1041 (5)
$DEVDEF	1	478 (1)	478 (1)
$DIBDEF	2	477 (1)	477 (1)
$DS_CNTRLC_S	1	925 (5)	1456 (6) 925 (5)
$DS_DSADEF	5	472 (1)	472 (1)
$DS_DSDEF	2	470 (1)	470 (1)
$DS_HPODEF	2	473 (1)	473 (1)
$DS_INITSCB_G	1	949 (5)	949 (5)
$DS_LOADPCS_S	1	942 (5)	942 (5)
$DS_SCBDEF	3	486 (1)	486 (1)
$DS_TYPEDEF	4	493 (1)	493 (1)
$EQU	1	494 (1)	470 (1) 488 (1) 492 (1) 493 (1) 494 (1)
$EQULS1	1	494 (1)	470 (1) 488 (1) 492 (1) 493 (1) 494 (1)
$EQULST	1	470 (1)	470 (1) 488 (1) 492 (1) 493 (1) 494 (1)
$EXIT_S	1	1907 (6)	1907 (6)
$FAB	4	584 (1)	584 (1) 590 (1)
$FABDEF	1	584 (1)	584 (1) 590 (1)
$GBLINI	2	470 (1)	470 (1) 488 (1) 489 (1) 492 (1) 493 (1) 494 (1)
$GETCHN_S	1	1079 (5)	1079 (5) 1110 (5) 1636 (6)
$GETSYI_S	1	1233 (5)	1233 (5)
$JIBDEF	3	480 (1)	480 (1)
$OFFDEF	1	719 (1)	719 (1)
$OPEN	1	1083 (5)	1083 (5)

```
$PCBDEF                          4        479    (1)      479    (1)
$PHDDEF                          6        476    (1)      476    (1)
$PRDEF                           4        482    (1)      482    (1)
$PRINT                           2       1369    (5)     1369    (5)     1428    (6)     1441    (6)     1445    (6)
$PRINTI_S                        1        937    (5)     1465    (6)      937    (5)
$PSLDEF                          2        483    (1)      483    (1)
$PUSHADR                         1        925    (5)     1037    (5)     1038    (5)     1039    (5)     1043    (5)
                                                         1079    (5)     1110    (5)     1116    (5)     1119    (5)
                                                         1120    (5)     1233    (5)     1456    (6)     1636    (6)
                                                         1639    (6)     1644    (6)     1647    (6)     1952    (6)
                                                          925    (5)      976    (5)
$PUSHTWO                         1       1116    (5)     1116    (5)     1644    (6)     1647    (6)
$QIOPUSH                         1       1039    (5)     1039    (5)     1116    (5)     1233    (5)     1644    (6)
                                                         1647    (6)
$QIOW_S                          1       1116    (5)     1116    (5)     1644    (6)     1647    (6)
$RAB                             2        587    (1)      587    (1)      593    (1)
$RABDEF                          1        587    (1)      587    (1)      593    (1)
$RMSCALL                         2       1083    (5)     1083    (5)     1085    (5)     1087    (5)     1089    (5)
$RPBDEF                          5        491    (1)      491    (1)
$SETAST_S                        1       1588    (6)     1588    (6)
$SETEXV_S                        1       1039    (5)     1039    (5)
$SHRDEF                          6        484    (1)      484    (1)
$SYIDEF                         18        485    (1)      485    (1)
$VIELD                           1                         489    (1)
$VIELD1                          1        494    (1)      489    (1)
BR_IF_NOT_APT                    1       1464    (6)     1464    (6)
BR_IF_NOT_LODFLG                 1       1877    (6)     1877    (6)
BR_IF_USER                       1       1898    (6)     1898    (6)
CLEAR_BINARY                     1       1423    (6)     1423    (6)
CLEAR_CRD_AUTOTEST_OFF           1       1347    (5)     1347    (5)     1426    (6)
CLEAR_CRD_MENUTEST_OFF           1       1345    (5)     1345    (5)     1424    (6)
CLEAR_CRD_MENUTEST_ON            1       1346    (5)     1346    (5)     1425    (6)
CLEAR_CRD_TRACE                  1       1348    (5)     1348    (5)
CLIDEF                           3        487    (1)      487    (1)
CMK                              1       1539    (6)     1539    (6)
CMKDEF                           1        488    (1)      488    (1)
DSFDEF                           3        489    (1)      489    (1)
DSQA                             2        492    (1)      492    (1)
ENVDEF                           1        490    (1)      490    (1)
ERRSUP_S                         1       1639    (6)     1639    (6)
MODNAM                           1        715    (1)      715    (1)
QA_MAIN                          1       1516    (6)     1516    (6)
SET_CRD_AUTOTEST_OFF             1       1382    (6)     1382    (6)
SET_CRD_MENUTEST_OFF             1       1395    (6)     1395    (6)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.13	00:00:00.30
Command processing	139	00:00:00.76	00:00:01.66
Pass 1	1922	00:00:33.53	00:00:47.19
Symbol table sort	13	00:00:02.86	00:00:03.28
Pass 2	964	00:00:07.52	00:00:12.47
Symbol table output	3	00:00:00.51	00:00:01.23

Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	223	00:00:02.96	00:00:11.85
Assembler run totals	3307	00:00:48.30	00:01:18.01

The working set limit was 1000 pages.
181816 bytes (356 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1853 non-local and 81 local symbols.
2178 source lines were read in Pass 1, producing 0 object records in Pass 2.
219 pages of virtual memory were used to define 81 macros.

+---------------------------+
! Macro library statistics !
+---------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	12
DRB1:[DS.WORK]DS.MLB;218	21
DRB1:[DS.WORK]CRD.MLB;12	2
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	39
TOTALS (all libraries)	77

2320 GETS were required to define 77 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) KERNEL/UPDA=(KERNEL.UPD,KERNEL.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA

Table of contents

(1) 113 Libraries, Equated Symbols
(1) 137 Work Psect Declarations
(1) 160 Data Psect Declarations
(1) 178 DSV$SETLOAD SET THE DEFAULT LOAD DEVICE/DIRECTORY
(1) 245 DSV$SHOWLOAD Display default load device
(1) 279 DSR$IFLOADDEV Is it the load device?
(1) 327 DSV$LOAD PROGRAM IMAGE LOAD SUBROUTINE

```
0000     1          .Title  LOAD    *** LOAD Set/Show load
0000     2          .Ident  /07-21/
0000     3          .NoShow Conditionals
0000     4 ;
0000     5 ; COPYRIGHT (C) 1977, 1980, 1982
0000     6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     7 ;
0000     8 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     9 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    10 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    11 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    12 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    13 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    14 ; REMAIN IN DEC.
0000    15 ;
0000    16 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    17 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    18 ; CORPORATION.
0000    19 ;
0000    20 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    22
0000    23 ;++
0000    24 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
0000    25 ;
0000    26 ; ABSTRACT:
0000    27 ;                    This routine is designed to read diagonostic
0000    28 ;          programs into memory.  It removes ALL breakpoints from
0000    29 ;          the previous program and opens the file specified by
0000    30 ;          the LOAD command and reads it starting at memory address
0000    31 ;          DS$GA_PRGBGN (^X200).
0000    32 ;
0000    33 ; ENVIRONMENT:
0000    34 ;          Loads programs in user mode
0000    35 ; and     Loads from floppy and RMS disks in stand-alone
0000    36 ;
0000    37 ; AUTHOR:        TOM SOUTTER     10-NOV-77       VERSION 01
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;                TOM SOUTTER     21-NOV-77       VERSION 02
0000    41 ;          01    SPR #16  "N.Y.I." MESSAGES FOR STAND-ALONE 'RUN' OR 'LOAD'.
0000    42 ;
0000    43 ;                TOM SOUTTER     24-MAR-78       VERSION 03 (ESSAA-4.01)
0000    44 ;          02    FIX TO LOAD ALGORITHM TO READ FILE BLOCKS "IN USE" ONLY
0000    45 ;
0000    46 ;                Roger Riggs     06-Jul-78       Version 04 (ESSAA-4.03)
0000    47 ;          03    Modify load sequence to use $READ and to read whole
0000    48 ;                file in one operation. verification is equivalent
0000    49 ;
0000    50 ;          04    Eliminate message for successful program load.  /TLS
0000    51 ;                Roger Riggs     26-Oct-78       Version 05 (ESSAA-5.01)
0000    52 ;          05    Add DSX$LOAD to allow diagnostic to read files into memory
0000    53 ;                Include facility for stand alone load from ODS-1/ODS-2 structures
0000    54 ;          06    Add floppy support in stand alone
0000    55 ;          07    Add SET LOAD and SHOW LOAD commands
0000    56 ;          08    Add DSR$IFLOADDEV to let attach(select) know if it is
0000    57 ;                dealing with the load device.
```

```
                                0000     58 ;        09      REMOVE W^ from calls to DSP$XX_READ
                                0000     59 ;
                                0000     60 ;                JOHN CIUKAJ     NOV-79
                                0000     61 ;        10      Modify DSX$LOAD for loadable drivers. BOO$QIO is used for
                                0000     62 ;                resident driver. Pseudo restart parameter block is developed
                                0000     63 ;                for use in BOO$QIO.
                                0000     64 ;
                                0000     65 ;        11      John Ciukaj     2-15-80
                                0000     66 ;                Change RPB block building in LOADFILE_SA to reflect new
                                0000     67 ;                structure of BOOTDRIVER module.
                                0000     68 ;                Roger Riggs     29-FEB-80
                                0000     69 ;        12      Added code to support 7th argument to DS$LOAD allowing
                                0000     70 ;                the program to read sections of the file, the LOAD$_LBN
                                0000     71 ;                argument supplies the starting logical block number of the
                                0000     72 ;                first block to be read from the file.
                                0000     73 ;                Roger Riggs, 5-June-1980, Version 5.4
                                0000     74 ;        13      Moved code to reset stacks and mode to before DS_CLEANUP is
                                0000     75 ;                called.
                                0000     76 ;
                                0000     77 ;                Dave Butenhof, 27-aug-1980
                                0000     78 ;        14      Define DEF$Q_DIR to be global, so it is accessible to
                                0000     79 ;                new standalone RMS routines.
                                0000     80 ;                Also deleted DSX$LOAD routine it has been moved to DSLOAD.
                                0000     81 ;                Roger Riggs, 17-Sep-1980, Version 6.0
                                0000     82 ;        15      Deleted Reference to L$A_LASTADR in DSV$LOAD, MAPFREE
                                0000     83 ;                now checks the loaded bit and L$A_LASTADR
                                0000     84 ;
                                0000     85 ;        16      - Jack Stansbury, 26-Oct-1981, Version 6.-
                                0000     86 ;                Added .LIBRARY statements for $DS and $DIAG.
                                0000     87 ;                Also fixed truncation errors.
                                0000     88 ;
                                0000     89 ;        17      Marion Baggett, 31-Mar-1982, Version 6.7
                                0000     90 ;                Added sys$library:lib .
                                0000     91 ;
                                0000     92 ;        18      Marion Baggett, 7-Apr-1982, Version 6.7
                                0000     93 ;                Added $DS_TYPEDEF to define type def codes.
                                0000     94 ;
                                0000     95 ;        19      Jack Stansbury, 13-April-1982, Version 6.7
                                0000     96 ;                Fixed truncation error.
                                0000     97 ;
                                0000     98 ;        20      Richard Brown, 15-June-82, Version 6.8
                                0000     99 ;                Clear image header storage buffer before loading diag.
                                0000    100 ;                Then save the name of the file being loaded, so the debugger
                                0000    101 ;                can use it later.
                                0000    102 ;                Also, clear debugger-resident flag before loading diag.
                                0000    103 ;
                                0000    104 ;        21      Richard Brown, 8-Mar-84, Version 6.15
                                0000    105 ;                Add code to clear all event flags before a new program
                                0000    106 ;                is loaded.
                                0000    107 ;
                                0000    108 ;        22      RE MUSE         15-MAY-84       VERSION 7.0
                                0000    109 ;                added support for longfile names
                                0000    110 ;
                                0000    111 ;--
```

```
                     0000   113          .SBTTL  Libraries, Equated Symbols
                     0000   114 ;
                     0000   115 ; INCLUDE FILES:
                     0000   116 ;
                     0000   117          .LIBRARY        /SYS$LIBRARY:LIB/                ;                              [17]
                     0000   118          .LIBRARY        /$DS/                            ;                              [16]
                     0000   119          .LIBRARY        /$DIAG/                          ;                              [16]
                     0000   120
                     0000   121 ;
                     0000   122 ; EQUATED SYMBOLS:
                     0000   123 ;
                     0000   124
                     0000   125          $DS_HDRDEF                          ; PROGRAM HEADER BLOCK SYMBOLS
                     0000   126          CLIDEF                              ; CLI COMMAND BLOCK OFFSETS
                     0000   127          DSFDEF                              ; CONTROL FLAG DEFINITIONS
                     0000   128          $DS_DSADEF                          ; CONTROL FLAG DEFINITIONS
                     0000   129          $DS_TYPEDEF                         ; DEFINE TYPE CODES                            [18]
                     0000   130
                     0000   131 ;Global References                                                                       [20]
                     0000   132
                     0000   133          .GLOBAL IMAGE_HEADER                ;IMAGE HEADER STORAGE BUFFER    [20]
                     0000   134          .GLOBAL IMAGE_HEADER_END            ;                               [20]
                     0000   135
```

```
                                        0000   137          .Subtitle       Work Psect Declarations
                                        0000   138 ;
                                        0000   139 ; OWN STORAGE:
                                        0000   140 ;
                                        0000   141
                                    00000000   142          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                        0000   143
                                    00000055   0000   144 DIAG_FILE_NAME::.BLKB 39+39+5+2          ; PLACE TO SAVE FILENAME OF LAST[22]
                                               0055   145                                           ; DIAGNOSTIC LOADED              [20]
                                               0055   146
                                               0055   147 SYS$SYSROOT::
59 53 24 53 59 53 0000005D'010E0000'           0055   148          .ASCID  "SYS$SYSROOT"            ; Device name to equate
                     54 4F 4F 52 53            0063
                                               0068   149 SYS$DISK::
49 44 24 53 59 53 00000070'010E0000'           0068   150          .ASCID  "SYS$DISK"               ; default Device name to equate
                                 4B 53         0076
                                               0078   151 DEF$Q_DEV::
                          00000080'00000000    0078   152          .LONG   0,10$                    ; Null default device
                                    0000009E   0080   153 10$:     .BLKB   30                       ; Storage for default device
                                               009E   154 DEF$Q_DIR::
41 4D 53 59 53 5B 000000A6'010E0000'           009E   155          .ASCID  "[SYSMAINT]"             ; Default load device
                                 5D 54 4E 49   00AC
                                    0000011F   00B0   156          .BLKB   <39*3>+4-10              ; Extra space for load device spec [22]
                                               011F   157
                                               011F   158          .ALIGN  LONG                     ; VMS REQUIREMENT
```

```
                                        0120   160          .Subtitle       Data Psect Declarations
                                        0120   161
                                    00000000   162          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                        0000   163
                                        0000   164 DEF_EXE:
30 3B 45 58 45 2E 00000008'010E0000'    0000   165          .ASCID  \.EXE;0\                ; Default for loaded files
                                        000E   166 FIL$GT_DDDEV::
                                     00'000E   167          .ASCIC  \\                      ; NO DEFAULT DEVICE
                                     00 000E
                                        000F   168 FIL$GT_DDSTRING::
   5D 54 4E 49 41 4D 53 59 53 5B 00'    000F   169          .ASCIC  \[SYSMAINT]\            ; DEFAULT DIRECTORY IN STAND ALONE
                                     0A 000F
                                        001A   170 T_SHOWLOAD:
         2F 21 53 41 21 53 41 21 00'    001A   171          .ASCIC  "!AS!AS!/"
                                     08 C01A
                                        0023   172 T_TRUNCATED:
61 6C 20 6F 6F 74 20 65 6C 69 46 00'    0023   173          .ASCIC  "File too large = !XL, max = !XL!/"
6D 20 2C 4C 58 21 20 3D 20 65 67 72     002F
      2F 21 4C 58 21 20 3D 20 78 61     003B
                                     21 0023
                                        0045   174 CLREF_ERROR_TEXT:                                          ; [21]
6C 63 20 72 6F 72 72 45 20 3F 3F 00'    0045   175          .ASCIC  "?? Error clearing event flags."       ; [21]
74 6E 65 76 65 20 67 6E 69 72 61 65     0051
               2E 73 67 61 6C 66 20     005D
                                     1E 0045
                                        0064   176
```

```
                     0064   178          .SBTTL  DSV$SETLOAD     SET THE DEFAULT LOAD DEVICE/DIRECTORY
                 00000000   179          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   180 ;++
                     0000   181 ; FUNCTIONAL DESCRIPTION:
                     0000   182 ;
                     0000   183 ;       This routine is called form CLI to process a "SET LOAD <filespec>"
                     0000   184 ;       command.  It saves the string pointed to by CLI$Q_FILE(R2) in
                     0000   185 ;       DEF$Q_DIR.
                     0000   186 ;
                     0000   187 ; CALLING SEQUCENCE:
                     0000   188 ;
                     0000   189 ;       BSBW    DSV$SETLOAD
                     0000   190 ;
                     0000   191 ; INPUT PARAMETERS:     NONE
                     0000   192 ;
                     0000   193 ; IMPLICIT INPUTS:
                     0000   194 ;
                     0000   195 ;       R2      Base of CLI DATA BASE
                     0000   196 ;       +       CLI DATA BASE
                     0000   197 ;
                     0000   198 ; OUTPUT PARAMETERS:    NONE
                     0000   199 ;
                     0000   200 ; IMPLICIT OUTPUTS:
                     0000   201 ;
                     0000   202 ;       DEF$Q_DIR gets descriptor of spec typed.
                     0000   203 ;               after spec is copied to local storage
                     0000   204 ;
                     0000   205 ; COMPLETION CODES:     N/A
```

```
                                0000   207 DSV$SETLOAD::
                          3C    BB 0000   208          PUSHR   #^M<R2,R3,R4,R5>        ; Save
                       50 05    D0 0002   209          MOVL    #5, R0                  ; Setup to clear 5 quadwords
                                   0005   210
                          7E    7C 0005   211 10$:     CLRQ    -(SP)                   ; Clear a quadword
                        FB 50   F5 0007   212          SOBGTR  R0, 10$                 ; Count and Branch if more
                                   000A   213
                          6E    7F 000A   214          PUSHAQ  (SP)                    ; Address of descriptors
                    7E    08    A2 000C   215          MOVQ    CLI$Q_FILE(R2), -(SP)   ; length, address of new default
            00000000'EF       03 FB 0010   216          CALLS   #3, BREAKUP_FILE        ; Merge in new fields
                                   0017   217
                       50       8E 7D 0017   218          MOVQ    (SP)+,R0                ; Get length/address of new device
                          10    13 001A   219          BEQL    20$                     ; Branch if none
            00000078'EF   50    D0 001C   220          MOVL    R0,DEF$Q_DEV            ; Set length of new device name
       54   00000078'EF   7D    7D 0023   221          MOVQ    DEF$Q_DEV,R4            ; Length/Address to save device
                          4C    10 002A   222          BSBB    90$                     ; Copy the device name
                                   002C   223
                       50       8E 7D 002C   224 20$:     MOVQ    (SP)+, R0               ; Length/addr of directory
                          10    13 002F   225          BEQL    30$                     ; Branch if none
            0000009E'EF   50    D0 0031   226          MOVL    R0,DEF$Q_DIR            ; Set new length of default directory
       54   0000009E'EF   7D    7D 0038   227          MOVQ    DEF$Q_DIR,R4            ; Length/Address to save directory
                          37    10 003F   228          BSBB    90$                     ; Copy directory name in
                                   0041   229
                                   0041   230 30$:     Br_If_Not_User  70$             ; Branch if not running in user-mode.   [18]
                                   0049   231
                                   0049   232          $CRELOG_S #2,SYS$DISK,DEF$Q_DEV ; Set new SYS$DISK
                          7E    7C 0060   233          CLRQ    -(SP)                   ; Last two parameters null
                0000009E'EF     9F 0062   234          PUSHAB  DEF$Q_DIR               ; Address of new default directory
            00000000'EF   03    FB 0068   235          CALLS   #3,SYS$SETDDIR          ; Set new default directory
                                   006F   236
                       5E    18 C0 006F   237 70$:     ADDL    #3*8, SP                ; Remove rest of parameters
                          3C    BA 0072   238          POPR    #^M<R2,R3,R4,R5>        ; Restore
                                05 0074   239          RSB
                                   0075   240
                       85    81 90 0075   241 80$:     MOVB    (R1)+, (R5)+            ; Copy a character
                          FA 50    F4 0078   242 90$:     SOBGEQ  R0,80$                  ; Count and copy if more
                                05 007B   243          RSB
```

```
                                007C    245          .SBTTL  DSV$SHOWLOAD    Display default load device
                                007C    246 ;++
                                007C    247 ; FUNCTIONAL DESCRIPTION:
                                007C    248 ;
                                007C    249 ;       This routine is called from CLI to display the current defaults
                                007C    250 ;       for device and directory.
                                007C    251 ;
                                007C    252 ; CALLING SEQUENCE:
                                007C    253 ;
                                007C    254 ;       BSBW    DSV$SHOWLOAD
                                007C    255 ;
                                007C    256 ; INPUT PARAMETERS:     NONE
                                007C    257 ;
                                007C    258 ; IMPLICIT INPUTS:
                                C07C    259 ;
                                007C    260 ;       R2      Address of CLI data base
                                007C    261 ;       +       contents of CLI$Q_FILE
                                007C    262 ;
                                007C    263 ; OUTPUT PARAMETERS:    NONE
                                007C    264 ;
                                007C    265 ; IMPLICIT OUTPUTS:     NONE
                                007C    266 ;
                                007C    267 ; COMPLETION CODES:     N/A
                                007C    268 ;--
                                007C    269
                                007C    270 DSV$SHOWLOAD::
          0000009E'EF   9F      007C    271          PUSHAB  L^DEF$Q_DIR                     ;                       [16]
          00000078'EF   7F      0082    272          PUSHAQ  L^DEF$Q_DEV                     ;                       [16]
          0000001A'EF   9F      0088    273          PUSHAB  L^T_SHOWLOAD                    ;                       [16]
              7E    01  9A      008E    274          movzbl  #DS$K_PRINTF,-(sp)              ;                       [17]
              7E    16  98      0091    275          cvtbl   #DS$K_TYPE_COMMAND_OUT,-(sp)    ;                       [17]
    00000000'EF   05  FB      0094    276          CALLS   #5,DSR$PRINT                    ;                       [17]
                      05      009B    277          RSB
```

```
                                009C   279          .SBTTL  DSR$IFLOADDEV   Is it the load device?
                                009C   280 ;++
                                009C   281 ; FUNCTIONAL DESCRIPTION:
                                009C   282 ;
                                009C   283 ;       This routine is called to determine if the device described
                                009C   284 ;       is the load device or not.
                                009C   285 ;
                                009C   286 ; CALLING SEQUENCE:
                                009C   287 ;
                                009C   288 ;       BSBW    DSR$IFLOADDEV
                                009C   289 ;
                                009C   290 ; INPUT PARAMETERS:     NONE
                                009C   291 ;
                                009C   292 ; IMPLICIT INPUTS:
                                009C   293 ;
                                009C   294 ;       R0      Length of device name
                                009C   295 ;       R1      Address of device name
                                009C   296 ;
                                009C   297 ; OUTPUT PARAMETERS:    NONE
                                009C   298 ;
                                009C   299 ; IMPLICIT OUTPUTS:     NONE
                                009C   300 ;
                                009C   301 ; COMPLETION CODES:
                                009C   302 ;
                                009C   303 ;       0       if device was not load device
                                009C   304 ;       1       If device was the load device
                                009C   305 ;
                                009C   306 ;--
                                009C   307
                                009C   308 DSR$IFLOADDEV::
                          3F    BB  009C   309          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save descriptor and regs
                54  00000078'  EF  7D  009E   310          MOVQ    L^DEF$Q_DEV,R4          ; Point to load device          [16]
                    00000000'  EF  16  00A5   311          Jsb     L^SCAN$DEVICE           ; Extract device name           [18]
                          0C    BA  00AB   312          POPR    #^M<R2,R3>              ; Restore desciptor desired
                          12    13  00AD   313          BEQL    40$                     ; Branch if no device present
                       52 50    B1  00AF   314          CMPW    R0,R2                   ; length match?
                          0D    12  00B2   315          BNEQ    40$                     ; Branch if not
                                00B4   316
                       83 81    91  00B4   317 20$:     CMPB    (R1)+,(R3)+             ; Character match?
                          08    12  00B7   318          BNEQ    40$                     ; Branch if not
                        F8 52   F5  00B9   319          SOBGTR  R2,20$                  ; Branch if more to compare
                       50 01    D0  00BC   320          MOVL    #1,R0                   ; Success
                          02    11  00BF   321          BRB     50$                     ; Restore regs and exit
                          50    D4  00C1   322 40$:     CLRL    R0                      ; Not load device
                                00C3   323
                          3C    BA  00C3   324 50$:     POPR    #^M<R2,R3,R4,R5>        ; Restore rest of registers
                                05  00C5   325          RSB
```

```
                        00C6    327          .SBTTL  DSV$LOAD        PROGRAM IMAGE LOAD SUBROUTINE
                       000000C6 328          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                        00C6    329 ;++
                        00C6    330 ; FUNCTIONAL DESCRIPTION:
                        00C6    331 ;
                        00C6    332 ;       READS AN PROGRAM IMAGE FILE ('.EXE/-HD') FROM A DISK
                        00C6    333 ;       INTO A MEMORY OVERLAY AREA DEFINED BY 'DS$A_PRGBGN' AND
                        00C6    334 ;       'DS$K_PRGSIZ'
                        00C6    335 ;
                        00C6    336 ; CALLING SEQUENCE:
                        00C6    337 ;
                        00C6    338 ;       BSBW    DSV$RUN                 TO LOAD THEN START
                        00C6    339 ;       BSBW    DSV$LOAD
                        00C6    340 ;
                        C0C6    341 ; INPUT PARAMETERS:     NONE
                        00C6    342 ;
                        00C6    343 ; IMPLICIT INPUTS:
                        00C6    344 ;
                        00C6    345 ;       CLI$Q_FILE = FILE SPEC (QUADWORD STRING DESCRIPTOR)
                        00C6    346 ;
                        00C6    347 ; OUTPUT PARAMETERS:    NONE
                        00C6    348 ;
                        00C6    349 ; IMPLICIT OUTPUTS:     NONE
                        00C6    350 ;
                        00C6    351 ; COMPLETION CODES:
                        00C6    352 ;
                        00C6    353 ;       R0      SS$_NORMAL      or failure  from DSX$LOAD
                        00C6    354 ;
                        00C6    355 ; SIDE EFFECTS:
                        00C6    356 ;
                        00C6    357 ;       PROGRAM AREA IS CLEARED BEFORE OVERLAY
                        00C6    358 ;           (BREAKPOINT TABLE MUST ALSO BE CLEARED)
                        00C6    359 ;--
```

```
                                00C6   361 DSV$RUN::
                             01 00C6   362         NOP                             ; 1 Inst so DSV$LOAD NEQ DSV$RUN
                                00C7   363
                                00C7   364 DSV$LOAD::
         00000000'EF       16 00C7   365         Jsb     SCRIPT$FLUSH            ; Flush scripts if console command      [18]
                                00CD   366
         00000000'EF       16 00CD   367         Jsb     L^INIT_CONTEXT          ; Reinit stacks, PCB...etc              [18]
   00000000'EF    01       E5 00D3   368         BBCC    #DS$V_[ODFLG,-
                             06 00DA   369                 L^DS$GL_FLAGS,5$        ; INDICATE NO IMAGE LOADED              [16]
         00000000'EF       16 00DB   370         Jsb     L^DS_CLEANUP            ; Cleanup if necessary                  [18]
                                00E1   371
                                00E1   372 5$:      Br_If_User      10$             ; Branch if in user-mode                [18]
   00000000'EF    6C       FA 00E9   373         CALLG   (AP),MAPFREE            ; Re-map free memory
                                00F0   374
                                00F0   375 10$:
   52   00000000'EF       9E 00F0   376         MOVAB   IMAGE_HEADER,R2         ; GET IMAGE HEADER STORAGE AREA ADDR.[20]
   53   00000000'EF       9E 00F7   377         MOVAB   IMAGE_HEADER_END,R3     ; END OF AREA                           [20]
             53    52       D1 00FE   378 100$:   CMPL    R2,R3                   ; WHILE NOT END OF BUFFER               [20]
                   04       13 0101   379         BEQL    200$                    ; DO                                    [20]
                   82       D4 0103   380         CLRL    (R2)+                   ;   CLEAR THE BUFFER                    [20]
                   F7       11 0105   381         BRB     100$                    ;   CHECK FOR DONE                      [20]
                            0107   382 200$:                                     ;                                       [20]
   52   00000000'EF       DE 0107   383         MOVAL   L^DS$GL_CLIBASE,R2      ; Reset R2 to point to cli database     [16]
      7E    0000'8F       3C 010E   384         MOVZWL  #DS$A_PRGBGN,-(SP)      ; Address to load file
      7E    0000'8F       3C 0113   385         MOVZWL  #DS$K_PRGSIZ,-(SP)      ; Length of load area
                            0118   386                                         ; NOW SAVE DIAG. FILENAME FOR DEBUGGER[20]
                   38       BB 0118   387         PUSHR   #^M<R3,R4,R5>           ; SAVE SOME REGISTERS                   [20]
         53    08 A2       DE 011A   388         MOVAL   CLI$Q_FILE(R2),R3       ; GET FILENAME DESCRIPTOR               [20]
   54   00000000'EF       DE 011E   389         MOVAL   DIAG_FILE_NAME,R4       ; GET STORAGE AREA                      [20]
            64    63       90 0125   390         MOVB    (R3),(R4)               ; GET CHAR. COUNT                       [20]
            53    04       C0 0128   391         ADDL2   #4,R3                   ; GET STRING POINTER                    [20]
            53    63       D0 012B   392         MOVL    (R3),R3                 ; GET STRING                            [20]
            55    84       90 012E   393         MOVB    (R4)+,R5                ; SET UP COUNTER OF CHARS TO MOVE[20]
                            0131   394 300$:                                     ; REPEAT                                [20]
            84    83       90 0131   395         MOVB    (R3)+,(R4)+             ;   MOVE A CHAR. TO SAVE AREA           [20]
                  55       97 0134   396         DECB    R5                      ;   COUNT A CHARACTER                   [20]
                  F9       14 0136   397         BGTR    300$                    ; UNTIL ENTIRE STRING COPIED            [20]
                  38       BA 0138   398         POPR    #^M<R3,R4,R5>           ; RESTORE SOME REGISTERS                [20]
      04000000 8F       CA 013A   399         BICL    #1@DSA$V_DEBUG,-        ; CLEAR THE DEBUGGER-RESIDENT           [20]
      0000FE00'EF               0140   400                 DSA$GL_FLAGS            ;   FLAG                              [20]
                                0145   401 ;
                                0145   402 ; Now clear all event flags.                                                  [21]
                                0145   403 ;
                  08       BB 0145   404         PUSHR   #^M<R3>                 ; Use R3 to hold EF Number              [21]
            53    01       D0 0147   405         MOVL    #1, R3                  ; Start with EFN 1; VDS uses 0          [21]
                  1C       E1 014A   406 400$:   BBC     #DSA$V_USER,-           ; IF user mode                          [21]
      0C 0000FE00'EF               014C   407                 DSA$GL_FLAGS,410$       ; THEN                              [21]
               18    53       D1 0152   408         CMPL    R3, #24                 ;   IF EFN >= 24                        [21]
                     07       19 0155   409         BLSS    410$                    ;   AND EFN <= 31                       [21]
               1F    53       D1 0157   410         CMPL    R3, #31                 ;   THEN                                [21]
                     02       14 015A   411         BGTR    410$                    ;       Skip this EFN                   [21]
                     0C       11 015C   412         BRB     420$                    ;                                       [21]
                     53       DD 015E   413 410$:   PUSHL   R3                      ; Put EFN on stack                      [21]
   00000000'EF       01       FB 0160   414         CALLS   #1, SYS$CLREF           ; Call service                          [21]
                  08    50    E9 0167   415         BLBC    R0, 425$                ; Branch on error                       [21]
            DC    53    3F    F3 016A   416 420$:   AOBLEQ  #63, R3, 400$          ; Next EFN                              [21]
                  08       BA 016E   417         POPR    #^M<R3>                 ; Restore R3                            [21]
```

```
                    17   11  0170   418          BRB     430$                            ; Continue                                [21]
                                0172   419
                                0172   420 425$:                                       ; Come here on $CLREF error               [21]
                    08   BA  0172   421          POPR    #^M<R3>                         ; Restore R3                              [21]
         00000045'EF   9F  0174   422          PUSHAB  CLREF_ERROR_TEXT                ; Put address of msg. on stack            [21]
               7E   01   9A  017A   423          movzbl  #DS$K_PRINTF,-(sp)              ;                                         [21]
               7E   16   98  017D   424          cvtbl   #DS$K_TYPE_COMMAND_OUT,-(sp);                                             [21]
    00000000'EF   03   FB  0180   425          CALLS   #3,DSR$PRINTF                   ; Print error message                     [21]
                    47   11  0187   426          BRB     DSV$LOAD_X                      ; And leave.                              [21]
                                0189   427
                                0189   428 430$:                                       ;                                         [21]
         00000000'EF   7F  0189   429          PUSHAQ  L^DEF_EXE                       ; Address of defaults for file                    [16]
               08 A2   7F  018F   430          PUSHAQ  CLI$Q_FILE(R2)                  ; Address of file descriptor
    00000000'EF   04   FB  0192   431          CALLS   #4,L^DSX$LOAD                   ; Load the file specified                 [16]
         00000000'EF   16   C199   432          Jsb     DSR$COMPLETION                  ; Type error text if necessary            [18]
                                019F   433
               2E 50   E9  019F   434 30$:     BLBC    R0,DSV$LOAD_X                   ; Branch if it failed
                                01A2   435          Set_LodFlg                      ; Indicate an image is loaded             [18]
          53   0200'8F   3C  01AA   436          MOVZWL  #DS$K_PRGSIZ+512,R3             ; Save max size
    51    00000214'EF   D0  01AF   437          MOVL    L$A_LASTADR,R1                  ; Get last address
               51   53   D1  01B6   438          CMPL    R3,R1                           ; Compare max to current
                    15   1E  01B9   439          BGEQU   DSV$LOAD_X                      ; Small enough, continue
                    0A   BB  01BB   440          PUSHR   #^M<R1,R3>                      ; Push Max size then current size
         00000023'EF   9F  01BD   441          PUSHAB  T_TRUNCATED                     ; Text of message
               7E   01   9A  01C3   442          movzbl  #DS$K_PRINTF,-(sp)              ;                                         [17]
               7E   16   98  01C6   443          cvtbl   #DS$K_TYPE_COMMAND_OUT,-(sp)    ;                                         [17]
    00000000'EF   05   FB  01C9   444          CALLS   #5,DSR$PRINT                    ; Tell the operator                       [17]
                                01D0   445
                                01D0   446 DSV$LOAD_X:
         00000000'EF   9F  01D0   447          PUSHAB  L^BEGIN                         ; Fake JSB for VRSTART or error           [18]
               0D 50   E9  01D6   448          BLBC    R0,10$                          ; If load failed, go back to BEGIN
000000C6'8F    04 A2   D1  01D9   449          CMPL    CLI$L_COMMAND(R2), -
                                01F1   450                  #DSV$RUN                ; Was it run command?
                    03   12  01E1   451          BNEQ    10$                             ; Branch if not run command
                 FE1A'   31  01E3   452          BRW     VRSTART                         ; Now issue START command
                    05   01E6   453 10$:     RSB                                     ; Return to BEGIN
                                01E7   454
                                01E7   455          .END
```

```
$$T1                            = 00000000   D
BEGIN                             ********   X   04
BIT...                          = 00000004   D
BREAKUP_FILE                      ********   X   04
CLI$K_BUFSIZ                    = 00000100   D
CLI$K_SIZE                        00000444   D
CLI$L_ADDRESS                     00000018   D
CLI$L_COMMAND                     00000004   D
CLI$L_DATA                        0000001C   D
CLI$L_FLAGS                       00000000   D
CLI$L_LAST                        00000024   D
CLI$L_NEXT                        00000030   D
CLI$L_PASS                        0000002C   D
CLI$L_SUBT                        00000028   D
CLI$L_TEST                        00000020   D
CLI$Q_BUFQWD                      00000034   D
CLI$Q_FILE                        00000008   D
CLI$Q_SECTION                     00000010   D
CLI$Q_TIME                        0000043C   D
CLI$T_BUFFER                      0000003C   D
CLI$V_ADAPTER                   = 00000018   D
CLI$V_ADR                       = 0000000B   D
CLI$V_ASCII                     = 00000013   D
CLI$V_BREAK                     = 0000000A   D
CLI$V_BRIEF                     = 00000018   D
CLI$V_BYTE                      = 0000000D   D
CLI$V_CLEAR                     = 00000002   D
CLI$V_DEC                       = 00000010   D
CLI$V_DEFAULT                   = 0000000C   D
CLI$V_DEPOSIT                   = 00000019   D
CLI$V_EVENT                     = 00000008   D
CLI$V_EXAM                      = 00000005   D
CLI$V_FLAGS                     = 00000009   D
CLI$V_HEX                       = 00000012   D
CLI$V_KERNEL                    = 00000017   D
CLI$V_LOAD                      = 00000006   D
CLI$V_LONG                      = 0000000F   D
CLI$V_NOTNUF                    = 00000001   D
CLI$V_OCT                       = 00000011   D
CLI$V_PREG                      = 0000001A   D
CLI$V_QA                        = 00000007   D
CLI$V_QACKLOOPLOOPS             = 0000001C   D
CLI$V_QAERRORPRINTS             = 0000001B   D
CLI$V_QAMULTIPLEPASS            = 0000001F   D
CLI$V_QASUBTESTLOOPS            = 0000001E   D
CLI$V_QATESTLOOPS               = 0000001D   D
CLI$V_REG                       = 00000014   D
CLI$V_REQUIRED                  = 00000000   D
CLI$V_RUN                       = 00000015   D
CLI$V_SET                       = 00000003   D
CLI$V_SHOW                      = 00000004   D
CLI$V_VALSEC                    = 00000016   D
CLI$V_WORD                      = 0000000E   D
CLREF_ERROR_TEXT                  00000045 R   D   03
DEF$Q_DEV                         00000078 RG  D   02
DEF$Q_DIR                         0000009E RG  D   02
DEF_EXE                           00000000 R   D   03
DIAG_FILE_NAME                    00000000 RG  D   02
DS$A_PRGBGN                       ********   X   04
DS$GL_CLIBASE                     ********   X   04
DS$GL_FLAGS                       ********   X   04
DS$K_PRGSIZ                       ********   X   04
DS$K_PRINTB                     = 00000002   D
DS$K_PRINTF                     = 00000001   D
DS$K_PRINTI                     = 00000000   D
DS$K_PRINTX                     = 00000003   D
DS$K_TYPE_ABORT_PROGRAM         = 00000014   D
DS$K_TYPE_ABORT_TEST            = 00000013   D
DS$K_TYPE_COMMAND_ERR           = 00000015   D
DS$K_TYPE_COMMAND_OUT           = 00000016   D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A   D
DS$K_TYPE_DS_PROMPT             = 00000001   D
DS$K_TYPE_DS_START              = 0000001D   D
DS$K_TYPE_ERRDEV                = 00000008   D
DS$K_TYPE_ERRHARD               = 00000006   D
DS$K_TYPE_ERROR_BODY            = 00000009   D
DS$K_TYPE_ERROR_END             = 0000000A   D
DS$K_TYPE_ERRPREP               = 0000001B   D
DS$K_TYPE_ERRSOFT               = 00000007   D
DS$K_TYPE_ERRSUP                = 00000004   D
DS$K_TYPE_ERRSYS                = 00000005   D
DS$K_TYPE_ERR_HALT              = 0000000D   D
DS$K_TYPE_EXCEPTION             = 0000000C   D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B   D
DS$K_TYPE_FIRST_PASS            = 00000011   D
DS$K_TYPE_GENERAL               = 00000000   D
DS$K_TYPE_GENERAL_ERROR         = 00000003   D
DS$K_TYPE_NO_TESTS              = 00000012   D
DS$K_TYPE_PARAM_ERROR           = 0000001C   D
DS$K_TYPE_PROGRAM_END           = 00000010   D
DS$K_TYPE_PROGRAM_INFO          = 00000017   D
DS$K_TYPE_PROGRAM_START         = 0000000F   D
DS$K_TYPE_QIO_INVADP            = 00000024   D
DS$K_TYPE_QIO_NODRIVER          = 00000022   D
DS$K_TYPE_QIO_WRONGVER          = 00000023   D
DS$K_TYPE_SCRIPT_ECHO           = 00000021   D
DS$K_TYPE_SCRIPT_PNF            = 0000001E   D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020   D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F   D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019   D
DS$K_TYPE_START_ERR             = 00000018   D
DS$K_TYPE_START_LIST            = 00000025   D
DS$K_TYPE_SUMMARY               = 0000000E   D
DS$K_TYPE_USER_PROMPT           = 00000002   D
DS$M_ABRTFLG                    = 00000040   D
DS$M_BADTIME                    = 00100000   D
DS$M_BATCH                      = 00400000   D
DS$M_BRKCLR                     = 00001000   D
DS$M_BRKPT                      = 00000800   D
DS$M_CHARFLG                    = 00000100   D
DS$M_CMDFLG                     = 00000080   D
DS$M_CTRLC                      = 00000001   D
DS$M_CTRLO                      = 00010000   D
DS$M_DEVFLG                     = 00000200   D
```

```
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_DEBUG                     = 0000001A     D
DSA$V_USER                      = 0000001C     D
DSR$COMPLETION                    ********     X    04
DSR$IFLOADDEV                     0000009C RG  D    04
DSV$LOAD                          000000C7 RG  D    04
DSV$LOAD_X                        000001D0 R   D    04
DSV$RUN                           000000C6 RG  D    04
DSV$SETLOAD                       00000000 RG  D    04
DSV$SHOWLOAD                      0000007C RG  D    04
DSX$LOAD                          ********     X    04
DSX$PRINT                         ********     X    04
DSX$PRINTF                        ********     X    04
DS_CLEANUP                        ********     X    04
FIL$GT_DDDEV                      0000000E RG  D    03
FIL$GT_DDSTRING                   0000000F RG  D    03
IMAGE_HEADER                      ********    GX    00
IMAGE_HEADER_END                  ********    GX    00
INIT_CONTEXT                      ********     X    04
L$A_CCP                           00000240     D
L$A_DEVP                          0000021C     D
L$A_DREG                          00000224     D
L$A_DTP                           00000218     D
L$A_ICP                           0000023C     D
L$A_LASTADR                       00000214     D
L$A_NAME                          00000208     D
L$A_REPP                          00000244     D
L$A_SECNAM                        00000250     D
L$A_SIATAB                        00000248     D
L$A_TSTCNT                        00000254     D
L$L_ENVIRON                       00000204     D
L$L_ERRTYP                        0000024C     D
L$L_HEADLENGTH                    00000200     D
L$L_REV                           0000020C     D
L$L_UNIT                          00000220     D
L$L_UNUSED                        00000228     D
L$L_UPDATE                        00000210     D
MAPFREE                           ********     X    04
SCAN$DEVICE                       ********     X    04
SCRIPT$FLUSH                      ********     X    04
SIZ...                          = 00000001     D
SYS$CLREF                         ********     X    04
SYS$CRELOG                        ********    GX    04
SYS$DISK                          00000068 RG  D    02
SYS$SETDDIR                       ********     X    04
SYS$SYSROOT                       00000055 RG  D    02
T_SHOWLOAD                        0000001A R   D    03
T_TRUNCATED                       00000023 R   D    03
VRSTART                           ********     X    04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000120 (288.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000064 (100.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	000001E7 (487.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T1	=00000000	232 (1)	232 (1)
BEGIN	00000000-XR		447 (1)
BIT...	=00000004	129 (1)	127 (1) 129 (1)
BREAKUP_FILE	00000000-XR		216 (1)
CLI$K_BUFSIZ	=00000100	126 (1)	126 (1)
CLI$K_SIZE	00000444	126 (1)	
CLI$L_ADDRESS	00000018	126 (1)	
CLI$L_COMMAND	00000C04	126 (1)	#-449 (1)
CLI$L_DATA	0000001C	126 (1)	
CLI$L_FLAGS	00000000	126 (1)	
CLI$L_LAST	00000024	126 (1)	
CLI$L_NEXT	00000030	126 (1)	
CLI$L_PASS	0000002C	126 (1)	
CLI$L_SUBT	00000028	126 (1)	
CLI$L_TEST	00000020	126 (1)	
CLI$Q_BUFQWD	00000034	126 (1)	
CLI$Q_FILE	00000008	126 (1)	#-215 (1) 388 (1) 430 (1)
CLI$Q_SECTION	00000010	126 (1)	
CLI$Q_TIME	0000043C	126 (1)	
CLI$T_BUFFER	0000003C	126 (1)	
CLI$V_ADAPTER	=00000018	126 (1)	
CLI$V_ADR	=0000000B	126 (1)	
CLI$V_ASCII	=00000013	126 (1)	
CLI$V_BREAK	=0000000A	126 (1)	
CLI$V_BRIEF	=0000001B	126 (1)	
CLI$V_BYTE	=0000000D	126 (1)	
CLI$V_CLEAR	=00000002	126 (1)	
CLI$V_DEC	=00000010	126 (1)	
CLI$V_DEFAULT	=0000000C	126 (1)	
CLI$V_DEPOSIT	=00000019	126 (1)	
CLI$V_EVENT	=00000008	126 (1)	
CLI$V_EXAM	=00000005	126 (1)	
CLI$V_FLAGS	=00000009	126 (1)	
CLI$V_HEX	=00000012	126 (1)	
CLI$V_KERNEL	=00000017	126 (1)	
CLI$V_LOAD	=00000006	126 (1)	
CLI$V_LONG	=0000000F	126 (1)	
CLI$V_NOTNUF	=00000001	126 (1)	
CLI$V_OCT	=00000011	126 (1)	
CLI$V_PREG	=0000001A	126 (1)	
CLI$V_QA	=00000007	126 (1)	
CLI$V_QACKLOOPLOOPS	=0000001C	126 (1)	
CLI$V_QAERRORPRINTS	=0000001B	126 (1)	
CLI$V_QAMULTIPLEPASS	=0000001F	126 (1)	
CLI$V_QASUBTESTLOOPS	=0000001E	126 (1)	
CLI$V_QATESTLOOPS	=0000001D	126 (1)	
CLI$V_REG	=00000014	126 (1)	
CLI$V_REQUIRED	=00000000	126 (1)	
CLI$V_RUN	=00000015	126 (1)	
CLI$V_SET	=00000003	126 (1)	

```
CLI$V_SHOW                      =00000004      126     (1)
CLI$V_VALSEC                    =00000016      126     (1)
CLI$V_WORD                      =0000000E      126     (1)
CLREF_ERROR_TEXT                 00000045-R    174     (1)     422     (1)
DEF$Q_DEV                        00000078-R    151     (1)   #-220     (1)     #-221     (1)       232     (1)       272     (1)
                                                             #-310     (1)
DEF$Q_DIR                        0000009E-R    154     (1)   #-226     (1)     #-227     (1)       234     (1)       271     (1)
DEF_EXE                          00000000-R    164     (1)     429     (1)
DIAG_FILE_NAME                   00000000-R    144     (1)     389     (1)
DS$A_PRGBGN                      00000000-XR                 #-384     (1)
DS$GL_CLIBASE                    00000000-XR                   383     (1)
DS$GL_FLAGS                      00000000-XR                   369     (1)       435     (1)
DS$K_PRGSIZ                      00000000-XR                 #-385     (1)     #-436     (1)
DS$K_PRINTB                     =00000002      129     (1)
DS$K_PRINTF                     =00000001      129     (1)   #-274     (1)     #-423     (1)     #-442     (1)
DS$K_PRINTI                     =00000000      129     (1)
DS$K_PRINTX                     =00000003      129     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014      129     (1)
DS$K_TYPE_ABORT_TEST            =00000013      129     (1)
DS$K_TYPE_COMMAND_ERR           =00000015      129     (1)
DS$K_TYPE_COMMAND_OUT           =00000016      129     (1)   #-275     (1)     #-424     (1)     #-443     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A      129     (1)
DS$K_TYPE_DS_PROMPT             =00000001      129     (1)
DS$K_TYPE_DS_START              =0000001D      129     (1)
DS$K_TYPE_ERRDEV                =00000008      129     (1)
DS$K_TYPE_ERRHARD               =00000006      129     (1)
DS$K_TYPE_ERROR_BODY            =00000009      129     (1)
DS$K_TYPE_ERROR_END             =0000000A      129     (1)
DS$K_TYPE_ERRPREP               =0000001B      129     (1)
DS$K_TYPE_ERRSOFT               =00000007      129     (1)
DS$K_TYPE_ERRSUP                =00000004      129     (1)
DS$K_TYPE_ERRSYS                =00000005      129     (1)
DS$K_TYPE_ERR_HALT              =0000000D      129     (1)
DS$K_TYPE_EXCEPTION             =0000000C      129     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      129     (1)
DS$K_TYPE_FIRST_PASS            =00000011      129     (1)
DS$K_TYPE_GENERAL               =00000000      129     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      129     (1)
DS$K_TYPE_NO_TESTS              =00000012      129     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      129     (1)
DS$K_TYPE_PROGRAM_END           =00000010      129     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      129     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      129     (1)
DS$K_TYPE_QIO_INVADP            =0000C024      129     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      129     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      129     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      129     (1)
DS$K_TYPE_SCRIPT_PNF            =0000C01E      129     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      129     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      129     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      129     (1)
DS$K_TYPE_START_ERR             =00000018      129     (1)
DS$K_TYPE_START_LIST            =00000025      129     (1)
DS$K_TYPE_SUMMARY               =0000000E      129     (1)
DS$K_TYPE_USER_PROMPT           =00000002      129     (1)
DS$M_ABRTFLG                    =00000040      127     (1)
DS$M_BADTIME                    =00100000      127     (1)
```

```
DS$M_BATCH                      =00400000     127     (1)
DS$M_BRKCLR                     =00001000     127     (1)
DS$M_BRKPT                      =00000800     127     (1)
DS$M_CHARFLG                    =00000100     127     (1)
DS$M_CMDFLG                     =00000080     127     (1)
DS$M_CTRLC                      =00000001     127     (1)
DS$M_CTRLO                      =00010000     127     (1)
DS$M_DEVFLG                     =00000200     127     (1)
DS$M_DISABLCC                   =01000000     127     (1)
DS$M_DONFLG                     =00002000     127     (1)
DS$M_ERRFLG                     =00000010     127     (1)
DS$M_EXCEPT                     =00080000     127     (1)
DS$M_EXETST                     =00040000     127     (1)
DS$M_HLTFLG                     =00000008     127     (1)
DS$M_LODFLG                     =00000002     127     (1)
DS$M_MEMMGT                     =00008000     127     (1)
DS$M_OUTPUT                     =00800000     127     (1)
DS$M_RUBFLG                     =00000020     127     (1)
DS$M_SCRIPT                     =00200000     127     (1)
DS$M_SETIMR                     =02000000     127     (1)
DS$M_STRFLG                     =00000004     127     (1)
DS$M_SUBT                       =00004000     127     (1)
DS$M_SYSFLG                     =00000400     127     (1)
DS$M_TIMRON                     =00020000     127     (1)
DS$V_ABRTFLG                    =00000006     127     (1)
DS$V_BADTIME                    =00000014     127     (1)
DS$V_BATCH                      =00000016     127     (1)
DS$V_BRKCLR                     =0000000C     127     (1)
DS$V_BRKPT                      =0000000B     127     (1)
DS$V_CHARFLG                    =00000008     127     (1)
DS$V_CMDFLG                     =00000007     127     (1)
DS$V_CTRLC                      =00000000     127     (1)
DS$V_CTRLO                      =00000010     127     (1)
DS$V_DEVFLG                     =00000009     127     (1)
DS$V_DISABLCC                   =00000018     127     (1)
DS$V_DONFLG                     =0000000D     127     (1)
DS$V_ERRFLG                     =00000004     127     (1)
DS$V_EXCEPT                     =00000013     127     (1)
DS$V_EXETST                     =00000012     127     (1)
DS$V_HLTFLG                     =00000003     127     (1)
DS$V_LODFLG                     =00000001     127     (1)     #-368   (1)     #-435   (1)
DS$V_MEMMGT                     =0000000F     127     (1)
DS$V_OUTPUT                     =00000017     127     (1)
DS$V_RUBFLG                     =00000005     127     (1)
DS$V_SCRIPT                     =00000015     127     (1)
DS$V_SETIMR                     =00000019     127     (1)
DS$V_STRFLG                     =00000002     127     (1)
DS$V_SUBT                       =0000000E     127     (1)
DS$V_SYSFLG                     =0000000A     127     (1)
DS$V_TIMRON                     =00000011     127     (1)
DSA$GL_FLAGS                     0000FE00                     230    (1)      372    (1)    #-400   (1)      407    (1)
DSA$V_DEBUG                     =0000001A                  #-399    (1)
DSA$V_USER                      =0000001C                  #-230    (1)    #-372    (1)    #-406   (1)
DSR$COMPLETION                   00000000-XR                432    (1)
DSR$IFLOADDEV                    0000009C-R    308     (1)
DSV$LOAD                         000000C7-R    364     (1)
DSV$LOAD_X                       000001D0-R    446     (1)     #-426   (1)     #-434   (1)     #-439   (1)
```

```
DSV$RUN                         000000C6-R   361     (1)     #-450     (1)
DSV$SETLOAD                     00000000-R   207     (1)
DSV$SHOWLOAD                    0000007C-R   270     (1)
DSX$LOAD                        00000000-XR                    431     (1)
DSX$PRINT                       00000000-XR                    276     (1)       444     (1)
DSX$PRINTF                      00000000-XR                    425     (1)
DS_CLEANUP                      00000000-XR                    370     (1)
FIL$GT_DDDEV                    0000000E-R   166     (1)
FIL$GT_DDSTRING                 0000000F-R   168     (1)
IMAGE_HEADER                    00000000-XR                    133     (1)       376     (1)
IMAGE_HEADER_END                00000000-XR                    134     (1)       377     (1)
INIT_CONTEXT                    00000000-XR                    367     (1)
L$A_LASTADR                     00000214                     #-437     (1)
MAPFREE                         00000000-XR                    373     (1)
SCAN$DEVICE                     00000000-XR                    311     (1)
SCRIPT$FLUSH                    00000000-XR                    365     (1)
SIZ...                          =00000001    127     (1)       127     (1)
SYS$CLREF                       00000000-XR                    414     (1)
SYS$CRELOG                      00000000-XR                    232     (1)
SYS$DISK                        00000068-R   149     (1)       232     (1)
SYS$SETDDIR                     00000000-XR                    235     (1)
SYS$SYSROOT                     00000055-R   147     (1)
T_SHOWLOAD                      0000001A-R   170     (1)       273     (1)
T_TRUNCATED                     00000023-R   172     (1)       441     (1)
VRSTART                         00000000-XR                  #-452     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ASNPUSH	1	232 (1)	232 (1)
$CRELOG_S	1	232 (1)	232 (1)
$DEF	1	129 (1)	
$DEFINI	1	125 (1)	125 (1) 128 (1)
$DS_DSADEF	5	128 (1)	128 (1)
$DS_HDRDEF	2	125 (1)	125 (1)
$DS_TYPEDEF	4	129 (1)	129 (1)
$EQU	1	129 (1)	129 (1)
$EQULS1	1	129 (1)	129 (1)
$EQULST	1	129 (1)	129 (1)
$GBLINI	2		127 (1) 129 (1)
$PUSHADR	1	232 (1)	232 (1)
$VIELD	1	127 (1)	127 (1)
$VIELD1	1	129 (1)	127 (1)
BR_IF_NOT_USER	1	230 (1)	230 (1)
BR_IF_USER	1	372 (1)	372 (1)
CLIDEF	3	126 (1)	126 (1)
DSFDEF	3	127 (1)	127 (1)
SET_LODFLG	1	435 (1)	435 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.11	00:00:00.28
Command processing	139	00:00:00.78	00:00:02.67
Pass 1	399	00:00:06.68	00:00:09.64
Symbol table sort	0	00:00:00.37	00:00:00.54
Pass 2	116	00:00:01.37	00:00:01.64
Symbol table output	26	00:00:00.19	00:00:00.44
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	40	00:00:00.71	00:00:00.71
Assembler run totals	766	00:00:10.25	00:00:15.97

The working set limit was 1000 pages.
32196 bytes (63 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 272 non-local and 23 local symbols.
455 source lines were read in Pass 1, producing 0 object records in Pass 2.
47 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	3
DRB1:[DS.WORK]DS.MLB;218	5
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	16

341 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) LOAD/UPDA=(LOAD.UPD,LOAD.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DI
```

Table of contents

(1) 54 LOAD MASSBUS ADAPTER MAP REGISTERS
(1) 100 LOAD UNIBUS ADAPTER MAP REGISTERS
(1) 177 GET PFN FROM INVALID PTE

```
                        0000     .1          .TITLE  LODMAP  *** LODMAP load adapter map registers
                        0000     .2          .IDENT  /05-02/
                        0000     .3          .LIST   MEB
                        0000     .4          .NLIST  CND
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
                        0000      7 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ; D. N. CUTLER 1-NOV-77
                        0000     27 ;
                        0000     28 ; LOAD MBA MAP REGISTERS
                        0000     29 ;
                        0000     30 ; MODIFIED BY:
                        0000     31 ;
                        0000     .1 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
                        0000     .2 ;
                        0000     .3 ; LOAD MBA AND UBA MAP REGISTERS
                        0000     .4 ;                Dave Butenhof   15-may-1980, Version 5.4
                        0000     .5 ;       02       Modify VMS V2.0 source to Supervisor Environment
                        0000     .6 ;
                        0000     32 ;       V04      RLR-TS04        Robert L. Rappaport     7-DEC-1979
                        0000     33 ;                Added an alternate entrypoint to IOC$LOADUBAMAP to
                        0000     34 ;                support byte aligned UNIBUS DMA devices
                        0000     35 ;
                        0000     36 ;       V03      NPKCOMET        N. KRONENBERG   11-JUN-1979
                        0000     37 ;                RETURNED IOC$LOADUBAMAP TO MODULE LOADMPEG.
                        0000     38 ;
                        0000     39 ;       V02      NPKCOMET        N. KRONENBERG   1-FEB-1979
                        0000     40 ;                REMOVED IOC$LOADUBAMAP TO MODULE LIOSUB.
                        0000     41 ;
```

```
                        0000     43 ;
                        0000     44 ; MACRO LIBRARY CALLS
                        0000     45 ;
                        0000     46
                        0000     .1
                        0000     .5
                        0000     47          $CRBDEF                         ;DEFINE CRB OFFSETS
                        0000     48          $MBADEF                         ;DEFINE MBA REGISTER OFFSET DEFINITIONS
                        0000     49          $PTEDEF                         ;DEFINE PAGE TABLE ENTRY FIELDS
                        0000     50          $UBADEF                         ;DEFINE UCB OFFSETS
                        0000     51          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000     52          $VECDEF                         ;DEFINE CRB TRANSFER VECTOR OFFSETS
```

```
                                00000000     .5          .PSECT  SEP, SHR, EXE, WRT, LONG
                                    0000     .6
-1                                  0000     54          .SBTTL  LOAD MASSBUS ADAPTER MAP REGISTERS
                                    0000     55 ;+
                                    0000     56 ; IOC$LOADMBAMAP - LOAD MASSBUS ADAPTER MAP REGISTERS
                                    0000     57 ;
                                    0000     58 ; THIS ROUTINE IS CALLED TO LOAD THE MASSBUS ADAPTER MAP REGISTERS, THE
                                    0000     59 ; BYTE COUNT REGISTER, AND THE VIRTUAL ADDRESS REGISTER.
                                    0000     60 ;
                                    0000     61 ; INPUTS:
                                    0000     62 ;
                                    0000     63 ;       R4 = ADDRESS OF MBA CONFIGURATION STATUS REGISTER.
                                    0000     64 ;       R5 = UCB ADDRESS OF UNIT TRANSFER IS TO OCCUR ON.
                                    0000     65 ;
                                    0000     66 ; OUTPUTS:
                                    0000     67 ;
                                    0000     68 ;       THE TRANSFER BYTE COUNT, STARTING PAGE OFFSET, AND ADDRESS OF THE
                                    0000     69 ;       PAGE TABLE ENTIRES THAT DESCRIBE THE TRANSFER ARE RETRIEVED FROM
                                    0000     70 ;       THE SPECIFED UCB AND USED TO LOAD THE MBA BYTE COUNT, VIRTUAL ADDRESS,
                                    0000     71 ;       AND MAP REGISTERS. ONE ADDITIONAL MAP REGISTER IS LOADED AS INVALID
                                    0000     72 ;       TO STOP THE TRANSFER IF A HARDWARE FAILURE SHOULD OCCUR.
                                    0000     73 ;
                                    0000     74 ;       R3 IS PRESERVED ACROSS CALL.
                                    0000     75 ;-
                                    0000     76
-1                                  0000     78 IOC$LOADMBAMAP::                             ;LOAD MASSBUS ADAPTER MAP REGISTERS
                          53    DD  0000     79          PUSHL   R3                          ;SAVE REGISTERS
                      52    6E  A5  3C  0002     80          MOVZWL  UCB$W_BCNT(R5),R2           ;GET TRANSFER BYTE COUNT
                      10  A4    52  CE  0006     81          MNEGL   R2,MBA$L_BCR(R4)            ;LOAD BYTE COUNT REGISTER
                      51    6C  A5  3C  000A     82          MOVZWL  UCB$W_BOFF(R5),R1           ;GET BYTE OFFSET IN PAGE
                      0C  A4    51  D0  000E     83          MOVL    R1,MBA$L_VAR(R4)            ;LOAD STARTING VIRTUAL ADDRESS
                      0C  A4    51  D0  0012     84          MOVL    R1,MBA$L_VAR(R4)            ;******TEMP UNTIL MBA ECO ******
                  52    01FF  C241  9E  0016     85          MOVAB   ^X1FF(R2)[R1],R2            ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
                      52    F7  8F  78  001C     86          ASHL    #-9,R2,R2                   ;CALCULATE NUMBER OF MAP REGISTERS TO LOAD
                                52      0020
                    51    0800  C4  DE  0021     87          MOVAL   MBA$L_MAP(R4),R1            ;GET ADDRESS OF MBA MAP REGISTERS
                      50    68  A5  D0  0026     88          MOVL    UCB$L_SVAPTE(R5),R0         ;GET ADDRESS OF PAGE TABLE
                          81    80  D0  002A     89 10$:      MOVL    (R0)+,(R1)+                 ;LOAD MAP REGISTER
                                09  18  002D     90          BGEQ    30$                         ;IF GEQ PTE INVALID
                            F8  52  F5  002F     91 20$:      SOBGTR  R2,10$                      ;ANY MORE TO LOAD?
                                61  D4  0032     92          CLRL    (R1)                        ;LOAD INVALID MAP ENTRY
                          53    8E  D0  0034     93          MOVL    (SP)+,R3                    ;RESTORE REGISTER
                                    05  0037     94          RSB                                 ;
                      53    FC  A0  D0  0038     95 30$:      MOVL    -4(R0),R3                   ;GET THE PTE (NOT FROM MAP REGISTER!)
                            009E  30  003C     96          BSBW    IOC$PTETOPFN                ;GET PFN FROM INVALID PTE
                  80000000  8F  C9  003F     97          BISL3   #^X80000000,R3,-4(R1)       ;AND LOAD THE MAP REGISTER
                    FC  A1    53      0045
                                E5  11  0048     98          BRB     20$                         ;
```

```
                                004A    100          .SBTTL  LOAD UNIBUS ADAPTER MAP REGISTERS
                                004A    101 ;+
                                004A    102 ; IOC$LOADUBAMAP  - LOAD UNIBUS ADAPTER MAP REGISTERS
                                004A    103 ; IOC$LOADUBAMAPA - LOAD UNIBUS ADAPTER MAP REGISTERS ALTERNATE ENTRY FOR
                                004A    104 ;       BYTE ALIGNED UNIBUS DMA DEVICES WHICH NEVER WISH TO SET THE BYTE
                                004A    105 ;       OFFSET BIT IN MAP REGISTERS.  IN ALL OTHER RESPECTS THESE TWO
                                004A    106 ;       ENTRYPOINTS PRODUCE IDENTICAL RESULTS.
                                004A    107 ;
                                004A    108 ; THIS ROUTINE IS CALLED TO LOAD THE UNIBUS ADAPTER MAP REGISTERS.
                                004A    109 ;
                                004A    110 ; INPUTS:
                                004A    111 ;
                                004A    112 ;       R5 = UCB ADDRESS OF UNIT TRANSFER IS TO OCCUR ON.
                                004A    113 ;
                                C04A    114 ;       IT IS ASSUMED THAT THE DATAPATH AND MAP REGISTERS HAVE BEEN PREVIOUSLY
                                004A    115 ;       ASSIGNED.
                                004A    116 ;
                                004A    117 ; OUTPUTS:
                                004A    118 ;
                                004A    119 ;       EACH MAP REGISTER IS LOADED WITH THE APPROPRIATE PAGE FRAME NUMBER
                                004A    120 ;       MERGED WITH THE DATAPATH DESIGNATOR AND BYTE OFFSET BIT. ONE ADDITIONAL
                                004A    121 ;       MAP REGISTER IS LOADED AS INVALID TO STOP THE TRANSFER IF A HARDWARE
                                004A    122 ;       FAILURE SHOULD OCCUR.
                                004A    123 ;
                                004A    124 ;       R3 IS PRESERVED ACROSS CALL.
                                004A    125 ;-
                                004A    126
                                004A    127          .ENABL  LSB
                                004A    128 IOC$LOADUBAMAPA::                   ;LOAD UNIBUS ADAPTER MAP REGISTERS - ALTERNATE
                                004A    129                                     ; HERE WE DUPLICATE THE CODE IN THE OTHER ENTRY
                                004A    130                                     ;  EXCEPT THAT WE DO NOT CHECK WHETHER THE BYTE
                                004A    131                                     ;  OFFSET IS ODD.  INSTEAD WE BRANCH DIRECTLY
                                004A    132                                     ;   PAST THE SETTING OF THE BYTE OFFSET BIT.
      7E    53    7D    004A    133          MOVQ    R3,-(SP)                   ;SAVE REGISTERS
  51    6C A5    3C    004D    134          MOVZWL  UCB$W_BOFF(R5),R1          ;GET BYTE OFFSET IN PAGE
  52    6E A5    3C    0051    135          MOVZWL  UCB$W_BCNT(R5),R2          ;GET TRANSFER BYTE COUNT
  53    20 A5    D0    0055    136          MOVL    UCB$L_CRB(R5),R3           ;GET ADDRESS OF CRB
            00 EF          0059    137          EXTZV   #VEC$V_DATAPATH,-          ;GET DATAPATH
            05             005B    138                  #VEC$S_DATAPATH,-          ; NUMBER
  54    27 A3          005C    139                  CRB$L_INTD+VEC$B_DATAPATH(R3),R4
            1B    11       005F    140          BRB     10$                        ; BRANCH AROUND TO JOIN COMMON CODE
                           0061    141 IOC$LOADUBAMAP::                    ;LOAD UNIBUS ADAPTER MAP REGISTERS
      7E    53    7D    0061    142          MOVQ    R3,-(SP)                   ;SAVE REGISTERS
  51    6C A5    3C    0064    143          MOVZWL  UCB$W_BOFF(R5),R1          ;GET BYTE OFFSET IN PAGE
  52    6E A5    3C    0068    144          MOVZWL  UCB$W_BCNT(R5),R2          ;GET TRANSFER BYTE COUNT
  53    20 A5    D0    006C    145          MOVL    UCB$L_CRB(R5),R3           ;GET ADDRESS OF CRB
            00 EF          0070    146          EXTZV   #VEC$V_DATAPATH,-          ;GET DATAPATH
            05             0072    147                  #VEC$S_DATAPATH,-          ; NUMBER
  54    27 A3          0073    148                  CRB$L_INTD+VEC$B_DATAPATH(R3),R4
        03 51    E9    0076    149          BLBC    R1,10$                     ;IF LBC WORD ALIGNED TRANSFER
     54    10    88    0079    150          BISB    #^X10,R4                   ;SET BYTE _FFSET BIT
  54    0400 8F    A8    007C    151 10$:     BISW    #^X400,R4                  ;MERGE VALID WITH BYTE OFFSET AND DATAPATH
            05    E1       0081    152          BBC     #VEC$V_LWAE,-              ;BRANCH IF LONGWORD ACCESS NOT ENABLED
      03 27 A3       0083    153                  CRB$L_INTD+VEC$B_DATAPATH(R3),15$
     54    20    88    0086    154          BISB    #^X20,R4                   ;ELSE SET LWAE FOR MAP REG
52    01FF C241    9E    0089    155 15$:     MOVAB   ^X1FF(R2)[R1],R2           ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
  52    F7 8F    78    008F    156          ASHL    #-9,R2,R2                  ;CALCULATE NUMBER OF MAP REGISTERS TO LOAD
```

```
                    52      0093
      26 A3         52  91  0094   157          CMPB    R2,CRB$L_INTD+VEC$B_NUMREG(R3) ;ENOUGH MAP REGISTERS ASSIGNED?
                    2C  1E  0098   158          BGEQU   40$                            ;IF GEQU NO
      51    28 B3   D0      009A   159          MOVL    @CRB$L_INTD+VEC$L_ADP(R3),R1 ;GET ADDRESS OF CONFIGURATION REGISTER
                    00  EF  009E   160          EXTZV   #VEC$V_MAPREG,-                ;GET STARTING REGISTER
                    0F          00A0   161                  #VEC$S_MAPREG,-
      50    24 A3         00A1   162                  CRB$L_INTD+VEC$W_MAPREG(R3),R0
   51   0800 C140   DE      00A4   163          MOVAL   UBA$L_MAP(R1)[R0],R1           ;GET ADDRESS OF FIRST MAP REGISTER TO LOAD
      50    68 A5   D0      00AA   164          MOVL    UCB$L_SVAPTE(R5),R0            ;GET ADDRESS OF PAGE TABLE
         53   80   D0      00AE   165 20$:      MOVL    (R0)+,R3                       ;GET NEXT PAGE TABLE ENTRY
                    02  19  00B1   166          BLSS    30$                            ;IF LSS VALID PAGE TABLE ENTRY
                    28  10  00B3   167          BSBB    IOC$PTETOPFN                   ;GFT PFN FROM INVALID PTE
   0B   15          54  F0  00B5   168 30$:      INSV    R4,#21,#11,R3                  ;INSERT VALID, BYTE OFFSET, AND DATAPATH
                    53          00B9
         81         53  D0  00BA   169          MOVL    R3,(R1)+                       ;LOAD UBA MAP REGISTER
            EE      52  F5  00BD   170          SOBGTR  R2,20$                         ;ANY MORE TO LOAD?
                    61  D4  00C0   171          CLRL    (R1)                           ;LOAD INVALID MAP ENTRY
         53   8E     7D  00C2   172          MOVQ    (SP)+,R3                       ;RESTORE REGISTERS
                    05  00C5   173          RSB                                    ;
                        00C6   174 40$:      BUG_CHECK UBMAPEXCED,FATAL             ;UNIBUS MAP REGISTER ALLOCATION EXCEEDED
    00000000'9F     16  00C6            JSB     @#BUG$CHECK
   50 41 4D 42 55 00'   00CC            .ASCIC  "UBMAPEXCED,FATAL"
   2C 44 45 43 58 45    00D2
      4C 41 54 41 46    00D8
                    10  00CC
                        00DD   175          .DSABL  LSB
```

```
                                00DD   177          .SBTTL  GET PFN FROM INVALID PTE
                                00DD   178 ;+
                                00DD   179 ; IOC$PTETOPFN - GET PFN FROM INVALID PTE
                                00DD   180 ;
                                00DD   181 ; THIS ROUTINE IS CALLED TO RETURN THE PAGE FRAME NUMBER FROM A
                                00DD   182 ; PAGE TABLE ENTRY WHICH HAS ALREADY BEEN DETERMINED TO BE NOT VALID.
                                00DD   183 ;
                                00DD   184 ; INPUTS:
                                00DD   185 ;
                                00DD   186 ;       R3 = PAGE TABLE ENTRY
                                00DD   187 ;
                                00DD   188 ; OUTPUTS:
                                00DD   189 ;
                                00DD   190 ;       R3 = PAGE FRAME NUMBER AND MAY INCLUDE THE FOLLOWING FIELDS
                                CODD   191 ;            VALID BIT, MODIFY BIT, PROTECTION FIELD, OWNER FIELD
                                00DD   192 ;
                                00DD   193 ;       ALL OTHER REGISTERS PRESERVED
                                00DD   194 ;-
                                00DD   195
                                00DD   196          .ENABL  LSB
-3                              00DD   200 IOC$PTETOPFN::
                 F8800000 8F    CA      00DD   201          BICL    #^C<PTE$M_TYP1 ! PTE$M_TYP0 !- ,PTE TYPE BITS
                          53            00E3
                                00E4   202                  PTE$M_GPTX>,R3          ;AND GPTX/PFN
                  01 53   1A    E0      00E4   203          BBS     #PTE$V_TYP1,R3,20$      ;BRANCH IF BAD PTE FOR I/O
-1                        05    00E8   205 10$:     RSB
                                00E9   206 20$:     BUG_CHECK INVPTEFMT,FATAL       ;INVALID PAGE TABLE ENTRY FORMAT
                00000000'9F     16      00E9                JSB     @#BUG$CHECK
              54 50 56 4E 49 00'        00EF                .ASCIC  "INVPTEFMT,FATAL"
              46 2C 54 4D 46 45         00F5
                    4C 41 54 41         00FB
                                0F      00EF
                                00FF   207          .DSABL  LSB
                                00FF   208
                                00FF   209          .END
```

```
BUG$CHECK           ********   X   02
CRB$B_MASK          0000000E     D
CRB$B_TYPE          0000000A     D
CRB$C_LENGTH        00000038     D
CRB$K_LENGTH        00000038     D
CRB$L_INTD          00000014     D
CRB$L_INTD2         00000038     D
CRB$L_LINK          00000010     D
CRB$L_WQBL          00000004     D
CRB$L_WQFL          00000000     D
CRB$W_REFC          0000000C     D
CRB$W_SIZE          00000008     D
IOC$LOADMBAMAP      00000000  RG  D  02
IOC$LOADUBAMAP      00000061  RG  D  02
IOC$LOADUBAMAPA     0000004A  RG  D  02
IOC$PTETOPFN        000000DD  RG  D  02
MBA$L_BCR         =  00000010     D
MBA$L_MAP         =  00000800     D
MBA$L_VAR         =  0000000C     D
PTE$M_GPTX        =  003FFFFF     D
PTE$M_TYP0        =  00400000     D
PTE$M_TYP1        =  04000000     D
PTE$V_TYP1        =  0000001A     D
SIZ...            =  00000001     D
UBA$L_MAP         =  00000800     D
UCB$B_AMOD          00000053     D
UCB$B_CEX           00000077     D
UCB$B_CM1           0000004A     D
UCB$B_CM2           0000004B     D
UCB$B_DEVCLASS      00000038     D
UCB$B_DEVTYPE       00000039     D
UCB$B_DIPL          00000052     D
UCB$B_DX_SCTCNT     000000A6     D
UCB$B_ERTCNT        00000070     D
UCB$B_ERTMAX        00000071     D
UCB$B_FEX           00000076     D
UCB$B_FIPL          0000000B     D
UCB$B_LOCSRV        0000003C     D
UCB$B_OFFNDX        00000094     D
UCB$B_OFFRTC        00000095     D
UCB$B_REMSRV        0000003D     D
UCB$B_SECTORS       0000003C     D
UCB$B_SLAVE         00000074     D
UCB$B_SPR           00000075     D
UCB$B_STATE         00000052     D
UCB$B_TRACKS        0000003D     D
UCB$B_TT_CRFILL     0000009D     D
UCB$B_TT_DECRF      000000A1     D
UCB$B_TT_DELFF      000000A2     D
UCB$B_TT_DESPEE     000000A0     D
UCB$B_TT_DETYPE     000000A4     D
UCB$B_TT_LFFILL     0000009E     D
UCB$B_TT_SPEED      0000009C     D
UCB$B_TYPE          0000000A     D
UCB$B_VERTSZ        0000003F     D
UCB$C_LENGTH        00000074     D
UCB$C_MB_LENGTH     00000090     D
UCB$C_TT_LENGTH     000000BC     D
UCB$K_LENGTH        00000074     D
UCB$K_MB_LENGTH     00000090     D
UCB$K_TT_LENGTH     000000BC     D
UCB$L_AMB           00000054     D
UCB$L_ASTQBL        00000010     D
UCB$L_ASTQFL        0000000C     D
UCB$L_CPID          0000005C     D
UCB$L_CRB           00000020     D
UCB$L_DDB           00000024     D
UCB$L_DEVCHAR       00000034     D
UCB$L_DEVDEPEND     0000003C     D
UCB$L_DPC           00000080     D
UCB$L_DUETIM        0000005C     D
UCB$L_DX_BFPNT      0000009C     D
UCB$L_DX_BUF        00000098     D
UCB$L_DX_RXDB       000000A0     D
UCB$L_EMB           00000078     D
UCB$L_FIRST         00000014     D
UCB$L_FPC           0000000C     D
UCB$L_FQBL          00000004     D
UCB$L_FQFL          00000000     D
UCB$L_FR3           00000010     D
UCB$L_FR4           00000014     D
UCB$L_IOQBL         00000044     D
UCB$L_IOQFL         00000040     D
UCB$L_IRP           0000004C     D
UCB$L_LINK          0000002C     D
UCB$L_LOGADR        00000064     D
UCB$L_MAXBLOCK      00000084     D
UCB$L_MB_MBX        0000007C     D
UCB$L_MB_PORT       0000008C     D
UCB$L_MB_RAST       00000078     D
UCB$L_MB_SHB        00000080     D
UCB$L_MB_WAST       00000074     D
UCB$L_MB_WIOQBL     00000088     D
UCB$L_MB_WIOQFL     00000084     D
UCB$L_MEDIA         0000008C     D
UCB$L_NT_DATSSB     00000074     D
UCB$L_NT_INTSSB     00000078     D
UCB$L_OPCNT         00000060     D
UCB$L_OWNUIC        0000001C     D
UCB$L_PID           00000028     D
UCB$L_RQBL          00000004     D
UCB$L_RQFL          00000000     D
UCB$L_SVAPTE        00000068     D
UCB$L_SVPN          00000064     D
UCB$L_TT_DECHAR     000000A8     D
UCB$L_TT_RDUE       0000008C     D
UCB$L_TT_RTIMOU     00000098     D
UCB$L_VCB           00000030     D
UCB$T_PARTNER       0000000C     D
UCB$W_BCNT          0000006E     D
UCB$W_BCR           00000096     D
UCB$W_BOFF          0000006C     D
UCB$W_BUFQUO        00000018     D
UCB$W_BYTESTOGO     0000003E     D
UCB$W_CHARGE        0000004A     D
UCB$W_CYLINDERS     0000003E     D
UCB$W_DA            0000008C     D
UCB$W_DC            0000008E     D
UCB$W_DEVBUFSIZ     0000003A     D
UCB$W_DEVSTS        0000005A     D
UCB$W_DIRSEQ        00000088     D
UCB$W_DSTADR        00000018     D
UCB$W_DX_BCR        000000A4     D
UCB$W_ECT           00000090     D
UCB$W_EC2           00000092     D
UCB$W_ERRCNT        00000072     D
UCB$W_FUNC          0000007E     D
UCB$W_MB_SEED       00000000     D
UCB$W_MSGCNT        00000016     D
UCB$W_MSGMAX        00000074     D
UCB$W_NT_CHAN       0000007C     D
UCB$W_OFFSET        0000008A     D
UCB$W_REFC          00000050     D
UCB$W_SIZE          00000008     D
UCB$W_SRCADDR       0000001A     D
UCB$W_STS           00000058     D
UCB$W_TT_DESIZE     000000A5     D
UCB$W_UNIT          00000048     D
UCB$W_VPROT         0000001A     D
VEC$B_DATAPATH      00000013     D
VEC$B_NUMREG        00000012     D
VEC$C_LENGTH        00000024     D
VEC$K_LENGTH        00000024     D
VEC$L_ADP           00000014     D
VEC$L_IDB           00000000     D
VEC$L_INITIAL       0000000C     D
VEC$L_START         0000001C     D
VEC$L_UNITDISC      00000020     D
VEC$L_UNITINIT      00000018     D
VEC$Q_DISPATCH      00000000     D
VEC$S_DATAPATH    =  00000005     D
VEC$S_MAPREG      =  0000000F     D
VEC$V_DATAPATH    =  00000000     D
VEC$V_LWAE        =  00000005     D
VEC$V_MAPREG      =  00000000     D
VEC$W_MAPREG         00000010     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	000000FF (255.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BUG$CHECK	00000000-XR		174 (1) 206 (1)
CRB$L_INTD	00000014		139 (1) 148 (1) 153 (1) #-157 (1) #-159 (1) 162 (1)
IOC$LOADMBAMAP	00000000-R	78 (1)	
IOC$LOADUBAMAP	00000061-R	141 (1)	
IOC$LOADUBAMAPA	0000004A-R	128 (1)	
IOC$PTETOPFN	000000DD-R	200 (1)	#-167 (1) #-96 (1)
MBA$L_BCR	=00000010		#-81 (1)
MBA$L_MAP	=00000800		87 (1)
MBA$L_VAR	=0000000C		#-83 (1) #-84 (1)
PTE$M_GPTX	=003FFFFF		#-202 (1)
PTE$M_TYP0	=00400000		#-201 (1)
PTE$M_TYP1	=04000000		#-201 (1)
PTE$V_TYP1	=0000001A		#-203 (1)
UBA$L_MAP	=00000800		163 (1)
UCB$L_CRB	00000020		#-136 (1) #-145 (1)
UCB$L_SVAPTE	00000068		#-164 (1) #-88 (1)
UCB$W_BCNT	0000006E		#-135 (1) #-144 (1) #-80 (1)
UCB$W_BOFF	0000006C		#-134 (1) #-143 (1) #-82 (1)
VEC$B_DATAPATH	00000013		139 (1) 148 (1) 153 (1)
VEC$B_NUMREG	00000012		#-157 (1)
VEC$L_ADP	00000014		#-159 (1)
VEC$S_DATAPATH	=00000005		#-138 (1) #-147 (1)
VEC$S_MAPREG	=0000000F		#-161 (1)
VEC$V_DATAPATH	=00000000		#-137 (1) #-146 (1)
VEC$V_LWAE	=00000005		#-152 (1)
VEC$V_MAPREG	=00000000		#-160 (1)
VEC$W_MAPREG	00000010		162 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CRBDEF	1	47 (1)	47 (1)
$DEFINI	1	47 (1)	47 (1) 48 (1) 49 (1) 50 (1) 51 (1) 52 (1)
$MBADEF	5	48 (1)	48 (1)
$PTEDEF	3	49 (1)	49 (1)
$UBADEF	6	50 (1)	50 (1)
$UCBDEF	10	51 (1)	51 (1)
$VECDEF	2	52 (1)	52 (1)
BUG_CHECK	1	174 (1)	174 (1) 206 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.31
Command processing	140	00:00:00.76	00:00:01.49
Pass 1	510	00:00:08.42	00:00:11.66
Symbol table sort	0	00:00:00.68	00:00:00.80
Pass 2	90	00:00:01.51	00:00:02.33
Symbol table output	13	00:00:00.10	00:00:00.10
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	15	00:00:00.17	00:00:00.24
Assembler run totals	807	00:00:11.79	00:00:16.95

The working set limit was 1000 pages.
40432 bytes (79 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 481 non-local and 10 local symbols.
222 source lines were read in Pass 1, producing 0 object records in Pass 2.
41 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	12

709 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) LODMAP/UPDA=(LODMAP.UPD,LODMAP.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 106 Declarations
(1) 147 DSX$Escape Routine
(1) 192 DSX$BgnSub Routine
(2) 298 DSX$EndSub Routine
(2) 488 DSX$CkLoop Routine
(3) 549 DSX$InLoop Routine

```
                                0000     1          .Title  LOOP    *** LOOP Loop control
                                0000     2          .Ident  /07-19/
                                0000     3          .NoShow Conditionals
                                0000     4
                                0000     5 ;
                                0000     6 ; Copyright (c) 1977, 1981, 1982
                                0000     7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                                0000     8 ;
                                0000     9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                                0000    10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                                0000    11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                                0000    12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                0000    13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                                0000    14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                                0000    15 ; REMAIN IN DEC.
                                0000    16 ;
                                0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                0000    18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                0000    19 ; CORPORATION.
                                0000    20 ;
                                0000    21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                                0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                0000    23
                                0000    24 ;++
                                0000    25 ; FACILITY:
                                0000    26 ;       VAX DIAGNOSTIC SUPERVISOR.
                                0000    27 ;
                                0000    28 ; ABSTRACT:
                                0000    29 ;
                                0000    30 ; ENVIRONMENT:
                                0000    31 ;
                                0000    32 ; AUTHOR:
                                0000    33 ;       KEN CHAPMAN     10-NOV-77       VERSION 01.
                                0000    34 ;
```

```
                                0000     36 ; MODIFICATIONS:
                                0000     37 ;
                                0000     38 ;                    9-Mar-1978      Roger Riggs
                                0000     39 ;            01      Fixed DS$INLOOP and DS$CKLOOP
                                0000     40 ;            02      Fixed BGNSUB and ENDSUB to use .entry and .vector
                                0000     41 ;                    also changed names to DSX$...
                                0000     42 ;                    12-Jun-1979     Roger Riggs
                                0000     43 ;            03      Turn off ^O before typing final loop pass message
                                0000     44 ;                    Dave Butenhof, 1-Mar-1980
                                0000     45 ;            04      Changed all references to APT mailbox to use longword
                                0000     46 ;                    offsets.
                                0000     47 ;                    Roger Riggs, 18-Mar-1980
                                0000     48 ;            05      Changed loop on subtest to execute the subtest the
                                0000     49 ;                    specified number of times instead of the previous +1.
                                0000     50 ;                    also made post loop on subtest processing call DS_CLEANUP
                                0000     51 ;                    - Dave Butenhof, 22-Jun-1981, version 6.4
                                0000     52 ;            06      Fix edit 5: delete explicit setting of DS$V_DONFLG, which
                                0000     53 ;                    prevented DS_CLEANUP from actually executing the diagnostic's
                                0000     54 ;                    cleanup code.
                                0000     55 ;            07      Fix /subtest to work right with /test:n:m.  It will only loop
                                0000     56 ;                    on the last test selected.
                                0000     57 ;
                                0000     58 ;            08      - Jack Stansbury, 21-Oct-1981, Version 6.-
                                0000     59 ;                    Added calls to QA$MAIN for the QA enhancement.
                                0000     60 ;                    Also added .LIBRARY statements for $DS and $DIAG.
                                0000     61 ;                    Also changed the case of some of the error messages.
                                0000     62 ;
                                0000     63 ;            09      - Dave Butenhof, 02-Nov-1981, version 6.-
                                0000     64 ;                    Fix truncation errors throughout.
                                0000     65 ;
                                0000     66 ;            10      - Jack Stansbury, 21-Nov-1981, Version 6.5
                                0000     67 ;                    Commented out a call to QA_MAIN because it appears that it
                                0000     68 ;                    will not be needed for the 6.5 DS version.
                                0000     69 ;
                                0000     70 ;            11      - Jack Stansbury, 13-Jan-1982, Version 6.6
                                0000     71 ;                    Added comments galore to make things easier to understand.
                                0000     72 ;
                                0000     73 ;            12      - Jack Stansbury, 14-Jan-1982, Version 6.6
                                0000     74 ;                    Added a call to the Summary Code routine in the EndSub routine.
                                0000     75 ;                    The diagnostic's summary code will now be executed when
                                0000     76 ;                    the requested number of loops on subtest have been done.
                                0000     77 ;                    This is more consistent with the EndPass routine.
                                0000     78 ;
                                0000     79 ;            13      - Jack Stansbury, 15-Jan-1982, Version 6.6
                                0000     80 ;                    Changed the FMTLOOPDON message to make it more like the
                                0000     81 ;                    EndPass message. Also added "Test x.y" to the message.
                                0000     82 ;
                                0000     83 ;            14      - Jack Stansbury, 20-Jan-1982, Version 6.6
                                0000     84 ;                    Changed the code in the EndSub routine to not print the
                                0000     85 ;                    header line when the Loop_On_Subtest QA check routine is
                                0000     86 ;                    executing. This is for QA.
                                0000     87 ;
                                0000     88 ;            15      - Jack Stansbury, 29-Mar-1982, Version 6.?
                                0000     89 ;                    Added some typecoding.
                                0000     90 ;
                                0000     91 ;            16      Marion Baggett, 30-mar-1982,    Version 6.7
                                0000     92 ;                    Changed all $DS_PRINT statements to the new form $PRINT.
```

```
0000     93 ;
0000     94 ;        17      Dave Butenhof, 08-Apr-1982, version 6.7
0000     95 ;                Fix truncation error.
0000     96 ;
0000     97 ;        18      Jack Stansbury, 13-April-1982, Version 6.7
0000     98 ;                Fix two truncation errors.
0000     99 ;
0000    100 ;        19      Peter Green, 18-JUL-1983, Version 6.12
0000    101 ;                Changed $DS_INLOOP to return correct status
0000    102 ;                returns, (DS$_ERROR or DS$_NORMAL)
0000    103 ;
0000    104 ;--
```

```
                        0000    106          .SBTTL  Declarations
                        0000    107 ;
                        0000    108 ; INCLUDE FILES:
                        0000    109 ;
                        0000    110          .LIBRARY        /SYS$LIBRARY:LIB/       ;                       [16]
                        0000    111          .LIBRARY        /$DS/                   ;                       [08]
                        0000    112          .LIBRARY        /$DIAG/                 ;                       [08]
                        0000    113
                        0000    114 ;
                        0000    115 ; MACROS:
                        0000    116 ;
                        0000    117
                        0000    118 ;
                        0000    119 ; EQUATED SYMBOLS:
                        0000    120 ;
                00000003 0000   121          BPTCODE = 03                    ; BREAKPOINT OPCODE
                        0000    122          $DS_DSADEF
                        0000    123          $DS_DSDEF
                        0000    124          $DS_HDRDEF
                        0000    125          APTDEF
                        0000    126          DSFDEF
                        0000    127          DSQA                            ; DSQA$K constant definitions           [08]
                        0000    128          DS_QADEFS                       ; Other QA$K constant definitions       [14]
                        0000    129          $DS_TypeDef                     ; Define the TypeCodes                  [15]
```

```
                                00000000   131          .PSECT  Data, Shr, NoExe, NoWrt, Byte
                                    0000   132
                                    0000   133 ;
                                    0000   134 ; OWN STORAGE:
                                    0000   135 ;
                                    0000   136
                                    0000   137          MODNAM  LOOP
                                    0005   138
                                    0005   139 FMTSEQERR:
6E 65 75 71 65 53 20 3F 3F 2F 21 00' 0005   140          .ASCIC  "!/?? Sequencing error. Should be in test !UW.!UW,"-    ;       [08]
20 2E 72 6F 72 72 65 20 67 6E 69 63  0011
6E 69 20 65 62 20 64 6C 75 6F 68 53  001D
55 21 2E 57 55 21 20 74 73 65 74 20  0029
74 20 64 65 68 63 61 65 72 20 2C 57  0035
2E 57 55 21 2E 57 55 21 20 74 73 65  C041
                           2F 21     004D
                              49     0005
                                    004F   141                  " reached test !UW.!UW.!/"                              ;       [08]
                                    004F   142 FMTLOOPDON:
66 6F 20 64 6E 45 20 2E 2E 2F 21 00' 004F   143          .ASCIC  "!/.. End of run, !UW error!%S detected, !UL loop!%S,"-  ;       [13]
72 65 20 57 55 21 20 2C 6E 75 72 20  005B
63 65 74 65 64 20 53 25 21 72 6F 72  0067
6F 6F 6C 20 4C 55 21 20 2C 64 65 74  0073
                     2C 53 25 21 70  007F
55 21 2E 4C 55 21 20 74 73 65 74 20  0084   144                  " test !UL.!UL,!/"-                                     ;       [13]
                        2F 21 2C 4C  0090
21 20 73 69 20 65 6D 69 74 20 20 20  0094   145                  "   time is !%D!/"                                      ;       [13]
                        2F 21 44 25  00A0
                              54     004F
```

```
                        00A4    147          .SBTTL  DSX$Escape Routine
                    00000000    148          .PSECT  Code, Shr, Exe, NoWrt, Byte
                        0000    149 ;++
                        0000    150 ; FUNCTIONAL DESCRIPTION:
                        0000    151 ;
                        0000    152 ;
                        0000    153 ; CALLING SEQUENCE:
                        0000    154 ;
                        0000    155 ;       $ESCAPE ARG
                        0000    156 ;
                        0000    157 ; INPUT PARAMETERS:
                        0000    158 ;
                        0000    159 ;       AP      = Address of where to escape
                        0000    160 ;
                        0000    161 ; IMPLICIT INPUTS:
                        0000    162 ;
                        0000    163 ;       DS$GL_FLAGS <DS$V_ERRFLG>
                        0000    164 ;
                        0000    165 ; OUTPUT PARAMETERS:
                        0000    166 ;
                        0000    167 ;       NONE
                        0000    168 ;
                        0000    169 ; IMPLICIT OUTPUTS:
                        0000    170 ;
                        0000    171 ;       NONE
                        0000    172 ;
                        0000    173 ; COMPLETION CODES:
                        0000    174 ;
                        0000    175 ;       NONE
                        0000    176 ;
                        0000    177 ; SIDE EFFECTS:
                        0000    178 ;
                        0000    179 ;       NONE
                        0000    180 ;
                        0000    181 ;--
```

```
                     0000  0000   183          .ENTRY  DSX$ESCAPE,^M<>
                           0002   184
          00000000'EF  16  0002   185          Jsb     L^KB_CHECK                ; ^C check.                                  [18]
                           0008   186          Br_If_Not_ErrFlg  REscapeX       ; Exit if no error has occurred              [18]
           10 AD   5C  D0  0010   187          MOVL    AP, 16(FP)                ; Replace return PC with ESCAPE address [11]
                           0014   188
                           0014   189 RESCAPEX:
                        04 0014   190          RET                               ; Return to diagnostic
```

```
                     0015   192          .SBTTL  DSX$BgnSub Routine
                     0015   193 ;++
                     0015   194 ; FUNCTIONAL DESCRIPTION:
                     0015   195 ;
                     0015   196 ;       This routine insures that the diagnostic program is sequencing
                     0015   197 ;       through its subtests in numerical order.  If an error is detected
                     0015   198 ;       the operator is notified and the command mode is entered as if
                     0015   199 ;       "Halt On Error" was selected. This means that the operator may
                     0015   200 ;       examine the general purpose registers and continue at the next
                     0015   201 ;       instruction of the program, if desired.
                     0015   202 ;
                     0015   203 ; CALLING SEQUENCE:
                     0015   204 ;
                     0015   205 ;       $BGNSUB
                     0015   206 ;               CALLG   $$$, @#DS$BGNSUB
                     0015   207 ;
                     0015   208 ; INPUT PARAMETERS:
                     0015   209 ;
                     0015   210 ;       4(AP)   = CURRENT TEST NUMBER (PROGRAM)
                     0015   211 ;       8(AP)   = NEW SUBTEST NUMBER (PROGRAM)
                     0015   212 ;
                     0015   213 ; IMPLICIT INPUTS:
                     0015   214 ;
                     0015   215 ;       DSA$GL_TESTNO           = CURRENT TEST NUMBER (SUPERVISOR).
                     0015   216 ;       DS$GL_FLAGS <DS$V_SUBT> = SUBTEST FLAG.
                     0015   217 ;
                     0015   218 ; OUTPUT PARAMETERS:
                     0015   219 ;
                     0015   220 ;       NONE
                     0015   221 ;
                     0015   222 ; IMPLICIT OUTPUTS:
                     0015   223 ;
                     0015   224 ;       DSA$GL_SUBTNO               = SUBTEST NUMBER.
                     0015   225 ;       DS$GA_LOOPADR               = LOOP ADDRESS.
                     0015   226 ;       DS$GL_FLAGS <DS$V_ERRFLG>   = ERROR FLAG.
                     0015   227 ;       DS$GL_FLAGS <DS$V_SUBT>     = SUBTEST FLAG.
                     0015   228 ;
                     0015   229 ; COMPLETION CODES:
                     0015   230 ;
                     0015   231 ;       NONE
                     0015   232 ;
                     0015   233 ; SIDE EFFECTS:
                     0015   234 ;
                     0015   235 ;       NONE
                     0015   236 ;
                     0015   237 ;--
```

```
                          0000  0015   239          .ENTRY  DSX$BGNSUB,^M<>
                                0017   240
                                0017   241  QA_MAIN Loop_DSX$BgnSub          ; Call GA$Main                                [08]
                                0020   242
        00000000'EF    0E   E2  0020   243  BBSS    #DS$V_SUBT, -            ; Set subtest flag. If it was already   [11]
                       1A       0027   244            L^DS$GL_FLAGS, 20$     ; set, print sequence error.            [11]
                                0028   245
                                0028   246  Clear_ErrFlg                     ; Clear the error flag.                 [18]
                                0030   247
                                0030   248  ;+
                                0030   249  ; Save the PC that is stored in the call frame. This PC is for the     [11]
                                0030   250  ; first instruction after the $DS_BgnSub routine. Save it in         [11]
                                0030   251  ; DS$GA_LoopAdr so that the EndSub routine can branch back to this   [11]
                                0030   252  ; address if it needs to (because of looping on subtest, etc.).      [11]
                                0030   253  ;-
                                0030   254
        00000000'EF    10 AD    D0  0030   255  MOVL    16 (FP), -               ; Save return address of this subtest.  [11]
                                0038   256            L^DS$GA_LOOPADR        ; LoopAdr is the address of where to    [11]
                                0038   257                                   ; loop back to from the EndSub routine. [11]
                                0038   258
        04 AC    0000FE50'EF    D1  0038   259  CMPL    L^DSA$GL_TESTNO, 4 (AP) ; Compare DS's TestNo with the progs.   [11]
                       62   13   0040   260  BEQL    50$                      ; Branch if they are the same (okay).   [11]
```

```
                                0042   262          ;+
                                0042   263          ; Sequencing error. The Supervisor's TestNo does not agree with the     [11]
                                0042   264          ; test number that the diagnostic passed to this routine (i.e., 16(AP)) [11]
                                0042   265          ; OR else the SubT flag was not set (indicating that two BgnSubs       [11]
                                0042   266          ; occurred without an intervening EndSub.                              [11]
                                0042   267          ;-
                                0042   268
                                0042   269 20$:     $Print  #DS$K_Type_Sequence_Error, -    ; There is a sequence error    [15]
                                0042   270                  #DS$K_PrintF, -            ; ... use printf                   [15]
                                0042   271                  L^FMTSEQERR, -             ; ... the format string            [15]
                                0042   272                  L^DSA$GL_TESTNO,-          ; ... rest of parameters           [15]
                                0042   273                  L^DSA$GL_SUBTNO,-
                                0042   274                  4(AP), 8(AP)
                                0067   275
                                C067   276          Set_Halt                           ; Set the HALT flag to halt on error.  [18]
     00000000'EF    07   E3     006F   277          BBCS    #DS$V_CMDFLG, -            ; Set the command flag. If not set,    [11]
                      11        0076   278                    L^DS$GL_FLAGS, 40$       ; branch (okay).                       [09]
                                0077   279          ERRSUP_S                           ; The command flag was already set.    [09]
                                0088   280
     00000000'EF    10 BD   DE 0088   281 40$:     MOVAL   @16(FP), L^DS$AA_BPTADDR; Save return PC.                         [09]
     00000000'EF    10 BD   90 0090   282          MOVB    @16(FP), L^DS$AB_BPTINST; Save the opcode located there.       [09]
              10 BD    03    90 0098   283          MOVB    #BPTCODE, @16(FP)          ; Store a breakpoint instruction at the [11]
                                009C   284                                             ; return location.                     [11]
     0000FE50'EF    04 AC   D0 009C   285          MOVL    4(AP), DSA$GL_TESTNO       ; Reset the test number.               [11]
```

```
                          00A4   287          ;+
                          00A4   288          ; Fall through to here if there was a sequencing error, or branch to    [11]
                          00A4   289          ; here if there was not a sequence error.                               [11]
                          00A4   290          ;-
                          00A4   291
     0000FE4C'EF    08 AC  D0   00A4   292 50$:     MOVL    8 (AP), DSA$GL_SUBTNO   ; Reset the subtest number.          [11]
           00000000'EF      16   00AC   293          Jsb     L^KB_CHECK              ; Check for TTY input.                [18]
                          00B2   294
                          00B2   295 RBGNSUBX:
                       04 00B2   296          RET                                 ; Return to next subtest.             [11]
```

```
00B3    298          .SBTTL  DSX$EndSub Routine
00B3    299 ;++
00B3    300 ; FUNCTIONAL DESCRIPTION:
00B3    301 ;
00B3    302 ;       The end of subtest routine checks both test and subtest numbers
00B3    303 ;       to assure that the program sequence is correct.
00B3    304 ;
00B3    305 ; CALLING SEQUENCE:
00B3    306 ;
00B3    307 ;       $ENDSUB
00B3    308 ;               CALLG   $$$, @#DS$ENDSUB
00B3    309 ;
00B3    310 ; INPUT PARAMETERS:
00B3    311 ;
00B3    312 ;       4(AP)   = CURRENT TEST NUMBER (PROGRAM).
00B3    313 ;       8(AP)   = CURRENT SUBTEST NUMBER (PROGRAM).
00B3    314 ;
00B3    315 ; IMPLICIT INPUTS:
00B3    316 ;
00B3    317 ;       DSA$GL_TESTNO               = TEST NUMBER (SUPERVISOR).
00B3    318 ;       DSA$GL_SUBTNO               = SUBTEST NUMBER (SUPERVISOR).
00B3    319 ;       DSA$GL_FLAGS <DSA$V_LOOP>   = LOOP FLAG.
00B3    320 ;       DS$GL_SUBTEST               = SUBTEST NUMBER FOR LOOPING.
00B3    321 ;       DS$GA_LOOPADR               = LOOP ADDRESS.
00B3    322 ;       DS$GL_FLAGS <DS$V_SUBT>     = SUBTEST FLAG.
00B3    323 ;       DS$GL_FLAGS <DS$V_ERRFLG>   = ERROR FLAG.
00B3    324 ;
00B3    325 ; OUTPUT PARAMETERS:
00B3    326 ;
00B3    327 ;       NONE
00B3    328 ;
00B3    329 ; IMPLICIT OUTPUTS:
00B3    330 ;
00B3    331 ;       DS$GL_FLAGS<DS$V_SUBT>  = SUBTEST FLAG.
00B3    332 ;
00B3    333 ; COMPLETION CODES:
00B3    334 ;
00B3    335 ;       NONE
00B3    336 ;
00B3    337 ; SIDE EFFECTS:
00B3    338 ;
00B3    339 ;       NONE
00B3    340 ;
00B3    341 ;--
```

```
                          0000  00B3   343          .ENTRY  DSX$ENDSUB,^M<>
                                00B5   344
                                00B5   345          QA_MAIN Loop_DSX$EndSub         ; Call QA$Main                          [08]
              00000000'EF   16  00BE   346          Jsb     L^KB_CHECK              ; Check for TTY input.                  [18]
                                00C4   347
                                00C4   348          ;+
                                00C4   349          ; Compare the Supervisor's test and subtest numbers with those of   [11]
                                00C4   350          ; the diagnostics (i.e., those that are passed). If they are not the [11]
                                00C4   351          ; same, it is a Sequence Error.                                     [11]
                                00C4   352          ;-
                                00C4   353
        04 AC   0000FE50'EF     D1  00C4   354          CMPL    L^DSA$GL_TESTNO, 4(AP)  ; Compare DS's TestNo with the progs.  [11]
                          0A    12  00CC   355          BNEQ    3$                      ; Branch if they are not equal.         [11]
        08 AC   0000FE4C'EF     D1  00CE   356          CMPL    L^DSA$GL_SUBTNO, 8(AP)  ; Compare DS's SubtNo with the progs.  [11]
                          03    13  00D6   357          BEQL    4$                      ; Branch if no error                    [11]
                                00D8   358
                        00DE    31  00D8   359  3$:     BRW     30$                     ; Branch if error in test or subtest
                                00DB   360
                                00DB   361          ;+
                                00DB   362          ; Check various flags.                                              [11]
                                00DB   363          ;-
                                00DB   364
                                00DB   365  4$:                                     ; TestNo and SubtNo are both okay.      [11]
           00000000'EF    0E    E5  00DB   366          BBCC    #DS$V_SUBT, -           ; Branch if there was no BgnSub.        [11]
                          50        00E2   367                    L^DS$GL_FLAGS, 17$  ;                                       [09]
           0000FE00'EF    02    E1  00E3   368          BBC     #DSA$V_LOOP, -          ; If Loop on Error flag not set,        [11]
                          08        00EA   369                    L^DSA$GL_FLAGS, 5$  ; branch.                           [11]
                                00EB   370          Br_If_Errflg    10$             ; If error occurred, branch.            [18]
```

```
                                00F3   372          ;+
                                00F3   373          ; Come to here if the Loop on Error flag is not set OR                  [11]
                                00F3   374          ;                 if no errors have occurred.                           [11]
                                00F3   375          ;-
                                00F3   376
00000000'EF   08 AC   D1        00F3   377 5$:      CMPL    8 (AP), L^DS$GL_SUBTEST ; Is this the subtest to loop on?       [11]
                 35   12        00FB   378          BNEQ    15$                     ; If not, branch.                       [11]
00000000'EF   04 AC   D1        00FD   379          CMPL    4 (AP), L^DS$GL_LSTTEST ; Are we on the correct test?           [09]
                 2B   12        0105   380          BNEQ    15$                     ; If not on correct test, exit.         [07]
                                0107   381
                                0107   382          ;+
                                0107   383          ; Add one to the PassNo count. Thus, keep track of the number of times  [11]
                                0107   384          ; this subtest has been executed. Execute it only 'x' times, where x is [11]
                                0107   385          ; /PASS:x.                                                              [11]
                                0107   386          ;-
                                0107   387
   50   0000FE54'EF   DE        0107   388          MOVAL   L^DSA$GL_PASSNO, R0     ; Point to pass number.
0C 60   0000FE08'EF   F3        010E   389          AOBLEQ  L^DSA$GL_PASSES, (R0), 10$ ; Count and branch if not done.
                                0116   390
                                0116   391          ;+
                                0116   392          ; Are all done the requested number of passes. However, if the pass     [11]
                                0116   393          ; request is 0, keep chugging forever.                                  [11]
                                0116   394          ;-
                                0116   395
         0000FE08'EF   D5       0116   396          TSTL    L^DSA$GL_PASSES         ; If pass count 0
                  04   13        011C   397          BEQL    10$                     ; Continue forever
                  60   D7        011E   398          DECL    (R0)                    ; Make true count of passes
                  14   11        0120   399          BRB     20$                     ; Branch to done code
```

B 1 UNLOCK AREAS IN IRPE'S
C 1 Symbol table
D 1 Symbol table
E 1 Symbol table
F 1 Symbol table
G 1 Psect synopsis
H 1 Cross reference
I 1 Cross reference
J 1 Cross reference
K 1 Cross reference
L 1 Cross reference
M 1 *** IOSNPG services for QIO
N 1 *** IOSNPG services for QIO
B 2 *** IOSNPG services for QIO
C 2 *** IOSNPG services for QIO
D 2 CANCEL I/O ON CHANNEL
E 2 FILL DIAGNOSTIC BUFFER
F 2 RELEASE I/O CHANNEL
G 2 REQUEST I/O CHANNEL
H 2 I/O REQUEST COMPLETION PROCESS
I 2 INITIATE I/O FUNCTION ON DEVIC
J 2 RELEASE BUFFERED DATAPATH
K 2 REQUEST BUFFERED DATAPATH
L 2 RELEASE UNIBUS MAP REGISTERS
M 2 REQUEST UNIBUS MAP REGISTERS
N 2 ALTER UBA MAP REGISTER BITMAP
B 3 SEARCH MAP REGISTER BITMAP AND
C 3 RETURN TO CALLER
D 3 WAITFOR INTERRUPT OR TIMEOUT A
E 3 WAITFOR INTERRUPT OR TIMEOUT A
F 3 ALLOCATE SYSTEM PAGE TABLE
G 3 Symbol table
H 3 Symbol table
I 3 Psect synopsis
J 3 Cross reference
K 3 Cross reference
L 3 Cross reference
M 3 Cross reference
N 3 *** IOSPGD services for QIO
B 4 *** IOSPGD services for QIO
C 4 *** IOSPGD services for QIO
D 4 CONVERT DEVICE NAME AND UNIT
E 4 FIND FREE I/O CHANNEL
F 4 SEARCH FOR DEVICE
G 4 SEARCH FOR DEVICE
H 4 SEARCH FOR DEVICE
I 4 UNLOCK I/O DATA BASE AND RETUR
J 4 VERIFY I/O CHANNEL NUMBER
K 4 Symbol table
L 4 Symbol table
M 4 Symbol table
N 4 Cross reference
B 5 Cross reference
C 5 Cross reference
D 5 *** IOSRAM random access I/O s
E 5 *** IOSRAM random access I/O s
F 5 *** IOSRAM random access I/O s
G 5 APPLY ECC CORRECTION
H 5 APPLY ECC CORRECTION
I 5 CONVERT LOGICAL BLOCK TO PHYSI
J 5 MAP VIRTUAL TO LOGICAL BLOCK
K 5 MAP VIRTUAL TO LOGICAL BLOCK
L 5 MAP VIRTUAL TO LOGICAL BLOCK
M 5 UPDATE TRANSFER PARAMETERS
N 5 SENSE DISK'S SIZE FDT ROUTINE
B 6 Symbol table
C 6 Symbol table
D 6 Cross reference
E 6 Cross reference
F 6 Cross reference
G 6 *** KERNEL Main control routin
H 6 *** KERNEL Main control routin
I 6 *** KERNEL Main control routin
J 6 *** KERNEL Main control routin
K 6 *** KERNEL Main control routin
L 6 *** KERNEL Main control routin
M 6 *** KERNEL Main control routin
N 6 *** KERNEL Main control routin
B 7 *** KERNEL Main control routin
C 7 *** KERNEL Main control routin
D 7 *** KERNEL Main control routin
E 7 *** KERNEL Main control routin
F 7 *** KERNEL Main control routin
G 7 *** KERNEL Main control routin
H 7 *** KERNEL Main control routin
I 7 Libraries and Macros
J 7 Equated Symbols
K 7 Equated Symbols
L 7 Data Psect Declarations
M 7 Work Psect Declarations
N 7 Work Psect Declarations
B 8 Work Psect Declarations
C 8 Data Psect Declarations
D 8 Data Psect Declarations
E 8 Data Psect Declarations
F 8 Data Psect Declarations
G 8 SHELL OF STAND ALONE SUPERVISO
H 8 SHELL OF STAND ALONE SUPERVISO
I 8 DIAGNOSTIC SUPERVISOR BOOTSTRA
J 8 DIAGNOSTIC SUPERVISOR BOOTSTRA
K 8 DIAGNOSTIC SUPERVISOR BOOTSTRA
L 8 DIAGNOSTIC SUPERVISOR BOOTSTRA
M 8 DIAGNOSTIC SUPERVISOR BOOTSTRA
N 8 USEP KERNEL ROUTINE
B 9 USEP KERNEL ROUTINE
C 9 USEP KERNEL ROUTINE
D 9 USEP KERNEL ROUTINE
E 9 USEP KERNEL ROUTINE
F 9 USEP KERNEL ROUTINE
G 9 USEP KERNEL ROUTINE
H 9 COMMON KERNEL REFRESH ENTRY PO
I 9 COMMON KERNEL REFRESH ENTRY PO
J 9 COMMON KERNEL REFRESH ENTRY PO
K 9 COMMON KERNEL REFRESH ENTRY PO
L 9 COMMON KERNEL REFRESH ENTRY PO
M 9 INIT_CONTEXT Initialize proces
N 9 INIT_CONTEXT Initialize proces
B 10 INIT_CONTEXT Initialize proces
C 10 INIT_CONTEXT Initialize proces
D 10 CONTROL-C AST HANDLER
E 10 KEYBOARD CHECK ROUTINE
F 10 KEYBOARD CHECK ROUTINE
G 10 KEYBOARD CHECK ROUTINE
H 10 KEYBOARD CHECK ROUTINE
I 10 DSX$CNTRLC Program enable for
J 10 DSV$EXIT, Process EXIT command
K 10 DSV$EXIT, Process EXIT command
L 10 EXIT$HANDLR Process EXIT handl
M 10 EXIT$HANDLR Process EXIT handl
N 10 DSR$Check_MenuTest_Off_Set Rou
B 11 DSR$Check_MenuTest_Off_Set Rou
C 11 DSR$Check_AutoTest_Off_Set Rou
D 11 DSR$Check_AutoTest_Off_Set Rou
E 11 DSR$Check_AutoTest_Off_Set Rou
F 11 DSR$Check_AutoTest_Off_Set Rou
G 11 DSX$GETTERM Get User Terminal
H 11 Symbol table
I 11 Symbol table
J 11 Symbol table
K 11 Symbol table
L 11 Symbol table
M 11 Symbol table
N 11 Symbol table
B 12 Psect synopsis
C 12 Cross reference
D 12 Cross reference
E 12 Cross reference
F 12 Cross reference
G 12 Cross reference
H 12 Cross reference
I 12 Cross reference
J 12 Cross reference
K 12 Cross reference
L 12 Cross reference
M 12 Cross reference
N 12 Cross reference
B 13 Cross reference
C 13 Cross reference
D 13 *** LOAD Set/Show load
E 13 *** LOAD Set/Show load
F 13 *** LOAD Set/Show load
G 13 Libraries, Equated Symbols
H 13 Work Psect Declarations
I 13 Data Psect Declarations
J 13 DSV$SETLOAD SET THE DEFAULT LO
K 13 DSV$SETLOAD SET THE DEFAULT LO
L 13 DSV$SHOWLOAD Display default l
M 13 DSR$IFLOADDEV Is it the load d
N 13 DSV$LOAD PROGRAM IMAGE LOAD SU
B 14 DSV$LOAD PROGRAM IMAGE LOAD SU
C 14 DSV$LOAD PROGRAM IMAGE LOAD SU
D 14 Symbol table
E 14 Symbol table
F 14 Psect synopsis
G 14 Cross reference
H 14 Cross reference
I 14 Cross reference
J 14 Cross reference
K 14 Cross reference
L 14 Cross reference

```
M 14  *** LODMAP load adapter map re
N 14  *** LODMAP load adapter map re
B 15  *** LODMAP load adapter map re
C 15  *** LODMAP load adapter map re
D 15  LOAD UNIBUS ADAPTER MAP REGIST
E 15  LOAD UNIBUS ADAPTER MAP REGIST
F 15  GET PFN FROM INVALID PTE
G 15  Symbol table
H 15  Psect synopsis
I 15  Cross reference
J 15  Cross reference
K 15  *** LOOP Loop control
L 15  *** LOOP Loop control
M 15  *** LOOP Loop control
N 15  *** LOOP Loop control
B 16  Declarations
C 16  Declarations
D 16  DSX$Escape Routine
E 16  DSX$Escape Routine
F 16  DSX$BgnSub Routine
G 16  DSX$BgnSub Routine
H 16  DSX$BgnSub Routine
I 16  DSX$BgnSub Routine
J 16  DSX$EndSub Routine
K 16  DSX$EndSub Routine
L 16  DSX$EndSub Routine
```

```
                        0122   401          ;+
                        0122   402          ; Come to here (if the Loop on Error flag is not set AND                [11]
                        0122   403          ;                 if an Error has occurred)                             [11]
                        0122   404          ;            OR (if the pass count has not reached its limit).          [11]
                        0122   405          ; Cause the loop to occur. That is, loop back to the instruction        [11]
                        0122   406          ; following the CALL to BgnSub.                                         [11]
                        0122   407          ;-
                        0122   408
      10 AD  00000000'EF  D0    0122   409 10$:    MOVL    L^DS$GA_LOOPADR, 16 (FP); Set PC to the saved LoopAdr.  [09]
                        012A   410          Set_SubT                          ; Set the SubT flag.                  [18]
                        0132   411
                     04 0132   412 15$:    RET                               ; Return with changed PC.             [11]
                        0133   413
                0083 31 0133   414 17$:    BRW     30$                       ; Branch on                           [09]
```

```
                                0136   416          ;+
                                0136   417          ; Loop on subtest w/pass count exhausted. This should mirror the        [11]
                                0136   418          ; DSX$EndPass routine.                                                  [11]
                                0136   419          ;-
                                0136   420
     50   00000000'EF   DE      0136   421 20$:     MOVAL   L^DS$GL_FLAGS,R0        ; Point to flags.                      [09]
60   01   07   01      F0      013D   422          INSV    #1, #DS$V_CMDFLG, #1, - ; Set Command mode.                    [06]
                                0142   423            (R0)                        ;                                      [11]
60   01   10   00      F0      0142   424          INSV    #0, #DS$V_CTRLO, #1, -  ; Clear Cntrl-O flag.                  [06]
                                0147   425            (R0)                        ;                                      [11]
                                0147   426
                                0147   427          Br_If_Not_QA    23$             ; If not running under QA, branch      [18]
                         03     E1     014F   428          BBC     #QA$K_LOOP_ON_SUBTEST, -; If not running Loop on Subtest check, [14]
                                0151   429            L^QA$AOB_CHECK_STATE,-; branch.                                [14]
                                C151   430            23$                         ;                                      [14]
      02 00000000'EF                    0157   431          BRB     24$             ; Otherwise, do not print ending line. [14]
                                0159   432
                                0159   433 23$:     $PRINT  #DS$K_TYPE_PROGRAM_END,- ; PRINT "End of run ...".             [16]
                                0159   434            #DS$K_PRINTF,-              ; Use printf.                          [16]
                                0159   435            L^FMTLOOPDON, -             ;                                      [14]
                                0159   436            L^DS$GL_ERRCNT, -           ; Current number of errors.            [09]
                                0159   437            L^DSA$GL_PASSNO, -          ; Number of loops done.                [11]
                                0159   438            L^DSA$GL_TESTNO, -          ; The test looped on.                  [13]
                                0159   439            L^DSA$GL_SUBTNO, -          ; The subtest looped on.               [13]
                                0159   440            #0                          ; Print current time.                  [11]
                                0186   441
          00000000'EF   16      0186   442 24$:     Jsb     L^INIT_CONTEXT          ; Reset stacks and processor mode.     [17]
                                018C   443          $DS_SUMMARY_S                   ; Exexcute the diagnostics summary code.[12]
          00000000'EF   16      0193   444          Jsb     L^DS_CLEANUP            ; Execute users cleanup code.
                                0199   445
                                0199   446          Clear_CtrlO                     ; Clear the Control-O flag. This makes [18]
                                01A1   447                                          ; ... sure it is off after the summary. [18]
0000FF40'EF   0A      D0        01A1   448          MOVL    #APM$_DONE, -           ; Tell APT we're all done.             [11]
                                01A8   449            L^DSA$GL_MSGTYP             ;                                      [11]
                                01A8   450          Br_If_Not_QA    25$             ; If not running under QA, branch.     [18]
                   FE4D'  31    01B0   451          BRW     VRRESTART               ; If running QA, branch to the START   [11]
                                01B3   452                                          ; routine. This should only happen on  [11]
                                01B3   453                                          ; the Loop_On_Subtest check routine.   [11]
          00000000'EF   16      01B3   454 25$:     Jsb     L^Begin                 ; Clean up the world. Go back to CLI.  [17]
```

```
                          01B9    456          ;+
                          01B9    457          ; Got here because not in subtest OR test or subtest numbers did not match.
                          01B9    458          ;-
                          01B9    459
                          01B9    460 30$:     $PRINT  #DS$K_TYPE_SEQUENCE_ERROR,-     ; PRINT "Sequencing error..."   [16]
                          01B9    461                    #DS$K_PRINTF,-               ; Use printf.                   [16]
                          01B9    462                    L^FMT$EQERR,-                ; ... the format string.        [18]
                          01B9    463                    L^DSA$GL_TESTNO,-            ; Current test number.          [11]
                          01B9    464                    L^DSA$GL_SUBTNO,-            ; Current subtest number.       [11]
                          01B9    465                    4(AP), 8(AP)                 ; User's test and subtest number. [11]
                          01DE    466
                          01DE    467          Set_Halt                              ; Set the Halt flag.            [18]
00000000'EF    07   E3    01E6    468          BBCS    #DS$V_CMDFLG, -               ; Set command flag.             [11]
                 11       01ED    469                  L^DS$GL_FLAGS, 45$           ;                               [09]
                          01EE    470          ERRSUP_S                              ; Supervisor error: command flag set. [11]
                          01FF    471
                          01FF    472          ;+
                          01FF    473          ; Save the return PC in BptAddr. Save the opcode located at the return  [11]
                          01FF    474          ; PC location. Store a BPT instruction at that location. Reset the      [11]
                          01FF    475          ; test and subtest numbers. Return to the diagnostic, and execute the   [11]
                          01FF    476          ; breakpoint instruction, causing the Cli to take over.                 [11]
                          01FF    477          ;-
                          01FF    478
      00000000'EF    10 BD   DE   01FF    479 45$:     MOVAL   @16 (FP), -           ; Save return PC.               [09]
                                  0207    480                    L^DS$AA_BPTADDR      ;                               [11]
      00000000'EF    10 BD   90   0207    481          MOVB    @16 (FP), -           ; Save the opcode located there. [09]
                                  020F    482                    L^DS$AB_BPTINST      ;                               [11]
                10 BD    03   90   020F    483          MOVB    #BPTCODE, @16 (FP)    ; Set a breakpoint at the return PC. [11]
0000FE50'EF     04 AC    D0   0213    484          MOVL    4 (AP), L^DSA$GL_TESTNO ; Set TestNo to the user's test number. [11]
0000FE4C'EF     08 AC    D0   021B    485          MOVL    8 (AP), L^DSA$GL_SUBTNO ; Set SubtNo to the user's subtest no. [11]
                         04   0223    486          RET                           ; Return to the diagnostic.     [11]
```

```
                     0224   488          .SBTTL  DSX$CkLoop Routine
                     0224   489 ;++
                     0224   490 ; FUNCTIONAL DESCRIPTION:
                     0224   491 ;
                     0224   492 ;       Check for loop flag AND an error flag, and loop if both
                     0224   493 ;       conditions are met.  Also keep track of previous loop conditions
                     0224   494 ;       and modify the loop if a new error occurs.
                     0224   495 ;
                     0224   496 ; CALLING SEQUENCE:
                     0224   497 ;
                     0224   498 ;       $CKLOOP ADR
                     0224   499 ;               CALLG   ADR, @#DS$CKLOOP
                     0224   500 ;
                     0224   501 ; INPUT PARAMETERS:
                     0224   502 ;
                     0224   503 ;       AP      = LOOP ADDRESS
                     0224   504 ;
                     0224   505 ; IMPLICIT INPUTS:
                     0224   506 ;
                     0224   507 ;       DS$GA_CHKLPPC                = CHECK LOOP PC.
                     0224   508 ;       DSA$GL_FLAGS <DSA$V_LOOP>    = LOOP FLAG.
                     0224   509 ;       DS$GL_FLAGS <DS$V_ERRFLG>    = ERROR FLAG.
                     0224   510 ;
                     0224   511 ; OUTPUT PARAMETERS:
                     0224   512 ;
                     0224   513 ;       NONE
                     0224   514 ;
                     0224   515 ; IMPLICIT OUTPUTS:
                     0224   516 ;
                     0224   517 ;       NONE
                     0224   518 ;
                     0224   519 ; COMPLETION CODES:
                     0224   520 ;
                     0224   521 ;       NONE
                     0224   522 ;
                     0224   523 ; SIDE EFFECTS:
                     0224   524 ;
                     0224   525 ;       NONE
                     0224   526 ;
                     0224   527 ;--
```

```
                        0000  0224   529          .ENTRY  DSX$CKLOOP, ^M<>
                              0226   530
                              0226   531          QA_MAIN Loop_DSX$CkLoop          ; Call QaMain.                                [08]
                              022F   532
                              022F   533          Jsb     L^KB_CHECK               ; Check for TTY input.                        [09]
          00000000'EF   16    022F   534          BBC     #DSA$V_LOOP, -           ; If not looping on error, branch.           [11]
     0000FE00'EF   02   E1    0235   535                    DSA$GL_FLAGS, RCKLOOPX
                   26         023C   536          Br_If_Not_ErrFlg  RCkLoopX       ; If no errors, branch.                       [18]
          00000000'EF   D5    0245   537          TSTL    L^DS$GA_CHKLPPC          ; Is this the first CHKLOOP after an          [09]
                              024B   538                                           ;   ERROR call?
                   0A   13    024B   539          BEQL    10$                      ; Branch if it is.                            [11]
 10 AD    00000000'EF   D1    024D   540          CMPL    L^DS$GA_CHKLPPC, 16(FP)  ; Is this the correct CHKLOOP.                [09]
                   0C   12    0255   541          BNEQ    RCKLOOPX                 ; Exit if not right call.                     [11]
                              0257   542
 00000000'EF   10 AD   D0    C257   543 10$:      MOVL    16(FP), L^DS$GA_CHKLPPC  ; Save PC of this CHKLOOP.                    [09]
          10 AD   5C   D0    025F   544          MOVL    AP, 16(FP)               ; Replace return PC with loop address.        [11]
                              0263   545
                              0263   546 RCKLOOPX:
                        04    0263   547          RET                              ; Return to the diagnostic.                   [11]
```

```
0264    549          .SBTTL  DSX$InLoop Routine
0264    550 ;++
0264    551 ; FUNCTIONAL DESCRIPTION:
0264    552 ;
0264    553 ;       This routine returns successfully if the diagnostic has
0264    554 ;       encountered an error AND the loop flag is set.
0264    555 ;
0264    556 ; CALLING SEQUENCE:
0264    557 ;
0264    558 ;       DS$INLOOP()
0264    559 ;
0264    560 ; INPUT PARAMETERS:     NONE
0264    561 ;
0264    562 ; IMPLICIT INPUTS:
0264    563 ;
0264    564 ;       DS$V_ERR in DS$GL_FLAGS
0264    565 ;       DSA$V_LOOP in DSA$GL_FLAGS
0264    566 ;
0264    567 ; OUTPUT PARAMETERS:    NONE
0264    568 ;
0264    569 ; IMPLICIT OUTPUTS:     NONE
0264    570 ;
0264    571 ; COMPLETION CODES:
0264    572 ;
0264    573 ;       R0=1 if looping and error else 0
0264    574 ;
0264    575 ; SIDE EFFECTS:         NONE
0264    576 ;--
```

```
                     0000  0264    578          .ENTRY  DSX$INLOOP,^M<>
                           0266    579
      00000000'EF     16   0266    580          Jsb     L^KB_CHECK                  ; Check for TTY input.
50    00660002 8F     D0   026C    581          MOVL    #DS$_ERROR,R0               ; CORRECT ERROR STATUS CODE          [19]
                           0273    582 ;        CLRL    R0                          ; Assume failure.                    [11]
                           0273    583          Br_If_Not_ErrFlg        10$        ; If error flag not set, branch.    [18]
0000FE00'EF   02   E1   0278    584          BBC     #DSA$V_LOOP, -                  ; If not looping on error, branch.   [11]
                07         0282    585                    L^DSA$GL_FLAGS, 10$
50    00660001 8F     D0   0283    586          MOVL    #DS$_NORMAL,R0              ; CORRECT NORMAL STATUS CODE         [19]
                           028A    56. ;        INCL    R0                          ; Error and LOOP on error set.       [11]
                           028A    588
                     04    028A    589 10$:     RET                                 ; Return to diagnostic.              [11]
                           028B    590
                           028B    591          .END
```

```
$$N                     =  00000005     D
$ER                     =  00000003     D
$MODULE                    00000000 R   D   02
APC$_ABORT              =  00000006     D
APC$_CONTINUE           =  00000009     D
APC$_DUMP               =  00000003     D
APC$_NOP                =  00000000     D
APC$_START              =  00000005     D
APC$_ZERO               =  00000004     D
APM$_ABTDON             =  00000006     D
APM$_DEVERR             =  00000002     D
APM$_DONE               =  0000000A     D
APM$_EXCEPT             =  0000000B     D
APM$_HARDERR            =  00000003     D
APM$_MORE               =  00000008     D
APM$_NOMES              =  00000000     D
APM$_PREPERR            =  0000000C     D
APM$_PRGERR             =  00000005     D
APM$_SOFTERR            =  00000004     D
APM$_SPOOL              =  00000009     D
APM$_SYSERR             =  00000001     D
BEGIN                      ********   X   03
BIT...                  =  00000004     D
BPTCODE                 =  00000003     D
DS$AA_BPTADDR              ********   X   03
DS$AB_BPTINST              ********   X   03
DS$GA_CHKLPPC              ********   X   03
DS$GA_LOOPADR              ********   X   03
DS$GL_ERRCNT               ********   X   03
DS$GL_FLAGS                ********   X   03
DS$GL_LSTTEST              ********   X   03
DS$GL_SUBTEST              ********   X   03
DS$K_BBC                =  0000001D   G D
DS$K_BBCC               =  00000021   G D
DS$K_BBCCI              =  00000023   G D
DS$K_BBCS               =  0000001F   G D
DS$K_BBS                =  0000001C   G D
DS$K_BBSC               =  00000020   G D
DS$K_BBSS               =  0000001E   G D
DS$K_BBSSI              =  00000022   G D
DS$K_BCC_M              =  0000001B   G D
DS$K_BCS_M              =  0000001A   G D
DS$K_BEQLU_B            =  00000005   G D
DS$K_BEQLU_M            =  00000011   G D
DS$K_BEQL_B             =  00000004   G D
DS$K_BEQL_M             =  00000010   G D
DS$K_BERROR             =  00000026   G D
DS$K_BGEQU_B            =  00000007   G D
DS$K_BGEQU_M            =  00000013   G D
DS$K_BGEQ_B             =  00000006   G D
DS$K_BGEQ_M             =  00000012   G D
DS$K_BGTRU_B            =  00000009   G D
DS$K_BGTRU_M            =  00000015   G D
DS$K_BGTR_B             =  00000008   G D
DS$K_BGTR_M             =  00000014   G D
DS$K_BLBC               =  00000025   G D
DS$K_BLBS               =  00000024   G D
DS$K_BLEQU_B            =  00000003   G D
DS$K_BLEQU_M            =  0000000F   G D
DS$K_BLEQ_B             =  00000002   G D
DS$K_BLEQ_M             =  0000000E   G D
DS$K_BLSSU_B            =  00000001   G D
DS$K_BLSSU_M            =  0000000D   G D
DS$K_BLSS_B             =  00000000   G D
DS$K_BLSS_M             =  0000000C   G D
DS$K_BNEQU_B            =  0000000B   G D
DS$K_BNEQU_M            =  00000017   G D
DS$K_BNEQ_B             =  0000000A   G D
DS$K_BNEQ_M             =  00000016   G D
DS$K_BNERROR            =  00000027   G D
DS$K_BVC_M              =  00000019   G D
DS$K_BVS_M              =  00000018   G D
DS$K_DS_STARTING_LOCATION =  00010000     D
DS$K_ERROR              =  00000002     D
DS$K_NORMAL             =  00000001     D
DS$K_PRINTB             =  00000002     D
DS$K_PRINTF             =  00000001     D
DS$K_PRINTI             =  00000000     D
DS$K_PRINTX             =  00000003     D
DS$K_SEVERE             =  00000004     D
DS$K_SUBSYS             =  00000066     D
DS$K_TYPE_ABORT_PROGRAM =  00000014     D
DS$K_TYPE_ABORT_TEST    =  00000013     D
DS$K_TYPE_COMMAND_ERR   =  00000015     D
DS$K_TYPE_COMMAND_OUT   =  00000016     D
DS$K_TYPE_CRD_AUTOTEST  =  0000001A     D
DS$K_TYPE_DS_PROMPT     =  00000001     D
DS$K_TYPE_DS_START      =  0000001D     D
DS$K_TYPE_ERRDEV        =  00000008     D
DS$K_TYPE_ERRHARD       =  00000006     D
DS$K_TYPE_ERROR_BODY    =  00000009     D
DS$K_TYPE_ERROR_END     =  0000000A     D
DS$K_TYPE_ERRPREP       =  0000001B     D
DS$K_TYPE_ERRSOFT       =  00000007     D
DS$K_TYPE_ERRSUP        =  00000C04     D
DS$K_TYPE_ERRSYS        =  00000005     D
DS$K_TYPE_ERR_HALT      =  0000000D     D
DS$K_TYPE_EXCEPTION     =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD =  0000000B     D
DS$K_TYPE_FIRST_PASS    =  00000011     D
DS$K_TYPE_GENERAL       =  00000000     D
DS$K_TYPE_GENERAL_ERROR =  00000003     D
DS$K_TYPE_NO_TESTS      =  00000012     D
DS$K_TYPE_PARAM_ERROR   =  0000001C     D
DS$K_TYPE_PROGRAM_END   =  00000010     D
DS$K_TYPE_PROGRAM_INFO  =  00000017     D
DS$K_TYPE_PROGRAM_START =  0000000F     D
DS$K_TYPE_QIO_INVADP    =  00000024     D
DS$K_TYPE_QIO_NODRIVER  =  00000022     D
DS$K_TYPE_QIO_WRONGVER  =  00000023     D
DS$K_TYPE_SCRIPT_ECHO   =  00000021     D
DS$K_TYPE_SCRIPT_PNF    =  0000001E     D
DS$K_TYPE_SCRIPT_PROMPT =  00000020     D
DS$K_TYPE_SCRIPT_SKIP   =  0000001F     D
```

```
DS$K_TYPE_SEQUENCE_ERROR        =  00000019     D
DS$K_TYPE_START_ERR             =  00000018     D
DS$K_TYPE_START_LIST            =  00000025     D
DS$K_TYPE_SUMMARY               =  0000000E     D
DS$K_TYPE_USER_PROMPT           =  00000002     D
DS$K_WARNING                    =  00000000     D
DS$M_ABRTFLG                    =  00000040     D
DS$M_BADTIME                    =  00100000     D
DS$M_BATCH                      =  00400000     D
DS$M_BRKCLR                     =  00001000     D
DS$M_BRKPT                      =  00000800     D
DS$M_CHARFLG                    =  00000100     D
DS$M_CMDFLG                     =  00000080     D
DS$M_CTRLC                      =  00000001     D
DS$M_CTRLO                      =  0001C000     D
DS$M_DEVFLG                     =  00000200     D
DS$M_DISABLCC                   =  01000000     D
DS$M_DONFLG                     =  00002000     D
DS$M_ERRFLG                     =  00000010     D
DS$M_EXCEPT                     =  00080000     D
DS$M_EXETST                     =  00040000     D
DS$M_HLTFLG                     =  00000008     D
DS$M_LODFLG                     =  00000002     D
DS$M_MEMMGT                     =  00008000     D
DS$M_OUTPUT                     =  00800000     D
DS$M_RUBFLG                     =  00000020     D
DS$M_SCRIPT                     =  00200000     D
DS$M_SETIMR                     =  02000000     D
DS$M_STRFLG                     =  00000004     D
DS$M_SUBT                       =  00004000     D
DS$M_SYSFLG                     =  00000400     D
DS$M_TIMRON                     =  00020000     D
DS$SUMMARY                      =  ********     X    03
DS$V_ABRTFLG                    =  00000006     D
DS$V_BADTIME                    =  00000014     D
DS$V_BATCH                      =  00000016     D
DS$V_BRKCLR                     =  0000000C     D
DS$V_BRKPT                      =  0000000B     D
DS$V_CHARFLG                    =  00000008     D
DS$V_CMDFLG                     =  00000007     D
DS$V_CTRLC                      =  00000000     D
DS$V_CTRLO                      =  00000010     D
DS$V_DEVFLG                     =  00000009     D
DS$V_DISABLCC                   =  00000018     D
DS$V_DONFLG                     =  0000000D     D
DS$V_ERRFLG                     =  00000004     D
DS$V_EXCEPT                     =  00000013     D
DS$V_EXETST                     =  0000C012     D
DS$V_HLTFLG                     =  00000003     D
DS$V_LODFLG                     =  00000001     D
DS$V_MEMMGT                     =  0000000F     D
DS$V_OUTPUT                     =  00000017     D
DS$V_RUBFLG                     =  00000005     D
DS$V_SCRIPT                     =  00000015     D
DS$V_SETIMR                     =  00000019     D
DS$V_STRFLG                     =  00000002     D
DS$V_SUBT                       =  0000000E     D
DS$V_SYSFLG                     =  0C000000A    D
DS$V_TIMRON                     =  00000011     D
DS$_ARITH                       =  00660000     D
DS$_ASBE                        =  00660118     D
DS$_BADLINK                     =  006600F0     D
DS$_BADTYPE                     =  006600E8     D
DS$_BIIC                        =  00660120     D
DS$_CHME                        =  006600A8     D
DS$_CHMK                        =  006600E0     D
DS$_DEVNAME                     =  00660108     D
DS$_ERROR                       =  00660002     D
DS$_FHWE                        =  00660068     D
DS$_FRAGBUF                     =  00660080     D
DS$_ICBUSY                      =  006600C8     D
DS$_ICERR                       =  006600C0     D
DS$_IHWE                        =  00660060     D
DS$_ILLCHAR                     =  00660018     D
DS$_ILLPAGCNT                   =  00660078     D
DS$_ILLUNIT                     =  00660100     D
DS$_INSFMEM                     =  00660050     D
DS$_IPL2HI                      =  006600B8     D
DS$_IVADDR                      =  00660040     D
DS$_IVVECT                      =  00660038     D
DS$_KRNLSTK                     =  00660090     D
DS$_LOGIC                       =  00660070     D
DS$_MCHK                        =  00660088     D
DS$_MMOFF                       =  00660058     D
DS$_NEEDUNIT                    =  006600F8     D
DS$_NODE                        =  00660128     D
DS$_NOPCS                       =  00660110     D
DS$_NORMAL                      =  00660001     D
DS$_NOSUPPORT                   =  006600B1     D
DS$_NOTDON                      =  00660030     D
DS$_NOTIMP                      =  006600B0     D
DS$_NULLSTR                     =  00660010     D
DS$_OVERFLOW                    =  00660008     D
DS$_POWER                       =  00660098     D
DS$_PROGERR                     =  00660020     D
DS$_SEVERE                      =  00660004     D
DS$_TRANSL                      =  006600A0     D
DS$_TRUNCATE                    =  00660028     D
DS$_UNEXPINT                    =  006600D8     D
DS$_VASFULL                     =  00660048     D
DS$_WARNING                     =  00660000     D
DSA$AL_APTMAIL                  =  0000FE00     D
DSA$AT_APTTXT                   =  0000FA00     D
DSA$GL_APTCOM                   =  0000FE04     D
DSA$GL_DEVLEN                   =  0000FE58     D
DSA$GL_ERRNO                    =  0000FE44     D
DSA$GL_EVENT                    =  0000FE48     D
DSA$GL_FLAGS                    =  0000FE00     D
DSA$GL_MSGTYP                   =  0000FE40     D
DSA$GL_PASSES                   =  0000FE08     D
DSA$GL_PASSNO                   =  0000FE54     D
DSA$GL_SECTNO                   =  0000FE10     D
DSA$GL_SID                      =  0000FE14     D
DSA$GL_SUBTNO                   =  0000FE4C     D
```

```
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0C00FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$V_HALT                    =   00000000    D
DSA$V_LOOP                    =   00C00002    D
DSA$V_QA                      =   0000000F    D
DSQA$K_DISPAT_AFTER_INIT      =   00000014    D
DSQA$K_DISPAT_BEFORE_INIT     =   00000013    D
DSQA$K_DISPAT_CALLTEST        =   00000015    D
DSQA$K_DISPAT_DONE_TESTS      =   00000018    D
DSQA$K_DISPAT_DSX$ENDPASS_BGN =   00000001    D
DSQA$K_DISPAT_DSX$ENDPASS_END =   00000002    D
DSQA$K_DISPAT_DSX$ENDPASS_MID =   00000000    D
DSQA$K_DISPAT_DS_CLEANUP_BGN  =   00000003    D
DSQA$K_DISPAT_DS_CLEANUP_END  =   00000004    D
DSQA$K_DUMMY_T                =   00000007    D
DSQA$K_ERROR_ERROR_BGN        =   00000006    D
DSQA$K_ERROR_ERROR_END        =   00000005    D
DSQA$K_KERNEL_INIT_CONTEXT    =   00000008    D
DSQA$K_LOOP_DSX$BGNSUB        =   00000009    D
DSQA$K_LOOP_DSX$CKLOOP        =   0000000B    D
DSQA$K_LOOP_DSX$ENDSUB        =   0000000A    D
DSQA$K_PARAM_DSX$GETADDRESS   =   0000000E    D
DSQA$K_PARAM_DSX$GETDATA      =   0000000C    D
DSQA$K_PARAM_DSX$GETLOGICAL   =   0000000F    D
DSQA$K_PARAM_DSX$GETSTRING    =   00000010    D
DSQA$K_PARAM_DSX$GETVIELD     =   0000000D    D
DSQA$K_QA_DSX$BRANCH          =   00000011    D
DSQA$K_START_BEFORE_HEADER    =   00000017    D
DSQA$K_START_VRRESTART        =   00000012    D
DSQA$K_START_VRSTART          =   00000016    D
DSX$BGNSUB                        00000015  RG  D  03
DSX$CKLOOP                        00000224  RG  D  03
DSX$ENDSUB                        000000B3  RG  D  03
DSX$ESCAPE                        00000000  RG  D  03
DSX$INLOOP                        00000264  RG  D  03
DSX$PRINT                         ********    X    03
DS_CLEANUP                        ********    X    03
DS_ERRSUP                         ********    X    03
FALSE                         =   00000000    D
FMTLOOPDON                        0000004F  R   D  02
FMTSEQERR                         00C00005  R   D  02
INIT_CONTEXT                      ********    X    03
KB_CHECK                          ********    X    03
L$A_CCP                           C0000240    D
L$A_DEVP                          0000021C    D
L$A_DREG                          00000224    D
L$A_DTP                           00000218    D
L$A_ICP                           0000023C    D
L$A_LASTADR                       00000214    D
L$A_NAME                          00000208    D
L$A_REPP                          00000244    D
L$A_SECNAM                        00000250    D
L$A_STATAB                        00000248    D
L$A_TSTCNT                        00000254    D
L$L_ENVIRON                       00000204    D
L$L_ERRTYP                        0000024C    D
L$L_HEADLENGTH                    00000200    D
L$L_REV                           0000020C    D
L$L_UNIT                          00000220    D
L$L_UNUSED                        00000228    D
L$L_UPDATE                        00000210    D
OFF                           =   00000000    D
ON                            =   00000001    D
QA$AOB_CHECK_STATE                ********    X    03
QA$K_ABORT_NOW                =   00000007    D
QA$K_APT_PHASE_ONE            =   0000000B    D
QA$K_APT_PHASE_TWO            =   0000000C    D
QA$K_BAD_ADDRESS              =   00000009    D
QA$K_CONTROL_C_CONT           =   0000000A    D
QA$K_DEALLOCATION             =   0000000E    D
QA$K_ERROR_PHASE_ONE          =   00000005    D
QA$K_ERROR_PHASE_THREE        =   00000007    D
QA$K_ERROR_PHASE_TWO          =   00000006    D
QA$K_FAILURE                  =   00000000    D
QA$K_FIRST_BRANCH_CODE        =   00000000    D
QA$K_FIRST_TIME               =   00000005    D
QA$K_INIT_CONTEXT_DONE        =   00000003    D
QA$K_LAST_BRANCH_CODE         =   00000025    D
QA$K_LOOP_ON_SUBTEST          =   00000003    D
QA$K_LOOP_ON_TEST             =   00000002    D
QA$K_LOOP_TEST_DONE           =   00000004    D
QA$K_MEMORY_MANAGE            =   0000000D    D
QA$K_MULTIPLE_PASS            =   00000001    D
QA$K_NODEF                    =   00000000    D
QA$K_NORMAL_START             =   00000000    D
QA$K_NO_HEADER                =   00000006    D
QA$K_NUMBER_OF_CHECKS         =   0000000F    D
QA$K_NUMBER_QA_FLAGS          =   00000008    D
QA$K_NUMB_ROUTINE_STATES      =   00000004    D
QA$K_QA_DEBUG                 =   00000001    D
QA$K_QA_DONE                  =   00000002    D
QA$K_RUN_BACKWARDS            =   00000004    D
QA$K_SECTION                  =   00000008    D
QA$K_SUCCESS                  =   00000001    D
QA$MAIN                           ********    X    03
QA$V_CHECK_DONE               =   00000003    D
QA$V_DOING_CLEANUP            =   00000002    D
QA$V_ERROR_FOUND              =   00000001    D
QA$V_INIT_DONE                =   00000000    D
RBGNSUBX                          000000B2  R   D  03
RCKLOOPX                          00000263  R   D  03
RESCAPEX                          00000014  R   D  03
SIZ...                        =   00000001    D
TRUE                          =   00000001    D
VRRESTART                         ********    X    03
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000000A4 (164.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	0000028B (651.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000005	465 (2)	274 (1) 440 (2) 465 (2)
$ER	=00000003	470 (2)	#-279 (1) #-470 (2)
$MODULE	00000000-R	137 (1)	279 (1) 470 (2)
APC$_ABORT	=00000006	125 (1)	
APC$_CONTINUE	=00000009	125 (1)	
APC$_DUMP	=00000003	125 (1)	
APC$_NOP	=00000000	125 (1)	
APC$_START	=00000005	125 (1)	
APC$_ZERO	=00000004	125 (1)	
APM$_ABTDON	=00000006	125 (1)	
APM$_DEVERR	=00000002	125 (1)	
APM$_DONE	=0000000A	125 (1)	#-448 (2)
APM$_EXCEPT	=0000000B	125 (1)	
APM$_HARDERR	=00000003	125 (1)	
APM$_MORE	=00000008	125 (1)	
APM$_NOMES	=00000000	125 (1)	
APM$_PREPERR	=0000000C	125 (1)	
APM$_PRGERR	=00000005	125 (1)	
APM$_SOFTERR	=00000004	125 (1)	
APM$_SPOOL	=00000009	125 (1)	
APM$_SYSERR	=00000001	125 (1)	
BEGIN	0000000C-XR		454 (2)
BIT...	=00000004	129 (1)	123 (1) 126 (1) 127 (1) 128 (1) 129 (1)
BPTCODE	=00000003	121 (1)	#-283 (1) #-483 (2)
DS$AA_BPTADDR	00000000-XR		#-281 (1) #-480 (2)
DS$AB_BPTINST	00000000-XR		#-282 (1) #-482 (2)
DS$GA_CHKLPPC	00000000-XR		#-537 (2) #-540 (2) #-543 (2)
DS$GA_LOOPADR	00000000-XR		#-256 (1) #-409 (2)
DS$GL_ERRCNT	00000000-XR		#-440 (2)
DS$GL_FLAGS	00000000-XR		186 (1) 244 (1) 246 (1) 278 (1) 367 (2) 370 (2) 410 (2) 421 (2) 446 (2) 469 (2) 536 (2) 583 (3)
DS$GL_LSTTEST	00000000-XR		#-379 (2)
DS$GL_SUBTEST	00000000-XR		#-377 (2)
DS$K_BBC	=0000001D	128 (1)	
DS$K_BBCC	=00000021	128 (1)	
DS$K_BBCCI	=00000023	128 (1)	
DS$K_BBCS	=0000001F	128 (1)	
DS$K_BBS	=0000001C	128 (1)	
DS$K_BBSC	=00000020	128 (1)	
DS$K_BBSS	=0000001E	128 (1)	
DS$K_BBSSI	=00000022	128 (1)	
DS$K_BCC_M	=0000001B	128 (1)	
DS$K_BCS_M	=0000001A	128 (1)	
DS$K_BEQLU_B	=00000005	128 (1)	
DS$K_BEQLU_M	=00000011	128 (1)	
DS$K_BEQL_B	=00000004	128 (1)	
DS$K_BEQL_M	=00000010	128 (1)	
DS$K_BERROR	=00000026	128 (1)	

DS$K_BGEQU_B	=00000007	128	(1)							
DS$K_BGEQU_M	=00000013	128	(1)							
DS$K_BGEQ_B	=00000006	128	(1)							
DS$K_BGEQ_M	=00000012	128	(1)							
DS$K_BGTRU_B	=00000009	128	(1)							
DS$K_BGTRU_M	=00000015	128	(1)							
DS$K_BGTR_B	=00000008	128	(1)							
DS$K_BGTR_M	=00000014	128	(1)							
DS$K_BLBC	=00000025	128	(1)	128	(1)					
DS$K_BLBS	=00000024	128	(1)							
DS$K_BLEQU_B	=00000003	128	(1)							
DS$K_BLEQU_M	=0000000F	128	(1)							
DS$K_BLEQ_B	=00000002	128	(1)							
DS$K_BLEQ_M	=0000000E	128	(1)							
DS$K_BLSSU_B	=00000C01	128	(1)							
DS$K_BLSSU_M	=0000000D	128	(1)							
DS$K_BLSS_B	=00000000	128	(1)	128	(1)					
DS$K_BLSS_M	=0000000C	128	(1)							
DS$K_BNEQU_B	=0000000B	128	(1)							
DS$K_BNEQU_M	=00000017	128	(1)							
DS$K_BNEQ_B	=0000000A	128	(1)							
DS$K_BNEQ_M	=00000016	128	(1)							
DS$K_BVERROR	=00000027	128	(1)							
DS$K_BVC_M	=00000019	128	(1)							
DS$K_BVS_M	=00000018	128	(1)							
DS$K_DS_STARTING_LOCATION	=00010000	123	(1)							
DS$K_ERROR	=00000002	123	(1)							
DS$K_NORMAL	=00000001	123	(1)							
DS$K_PRINTB	=00000002	129	(1)							
DS$K_PRINTF	=00000001	129	(1)	#-274	(1)	#-440	(2)	#-465	(2)	
DS$K_PRINTI	=00000000	129	(1)							
DS$K_PRINTX	=00000003	129	(1)							
DS$K_SEVERE	=00000004	123	(1)							
DS$K_SUBSYS	=00000066	123	(1)	123	(1)					
DS$K_TYPE_ABORT_PROGRAM	=00000014	129	(1)							
DS$K_TYPE_ABORT_TEST	=00000013	129	(1)							
DS$K_TYPE_COMMAND_ERR	=00000015	129	(1)							
DS$K_TYPE_COMMAND_OUT	=00000016	129	(1)							
DS$K_TYPE_CRD_AUTOTEST	=0000001A	129	(1)							
DS$K_TYPE_DS_PROMPT	=00000001	129	(1)							
DS$K_TYPE_DS_START	=0000001D	129	(1)							
DS$K_TYPE_ERRDEV	=00000008	129	(1)							
DS$K_TYPE_ERRHARD	=00000006	129	(1)							
DS$K_TYPE_ERROR_BODY	=00000009	129	(1)							
DS$K_TYPE_ERROR_END	=0000000A	129	(1)							
DS$K_TYPE_ERRPREP	=0000001B	129	(1)							
DS$K_TYPE_ERRSOFT	=00000007	129	(1)							
DS$K_TYPE_ERRSUP	=00000004	129	(1)							
DS$K_TYPE_ERRSYS	=00000005	129	(1)							
DS$K_TYPE_ERR_HALT	=0000000D	129	(1)							
DS$K_TYPE_EXCEPTION	=0000000C	129	(1)							
DS$K_TYPE_EXCEPTION_HEAD	=0000000B	129	(1)							
DS$K_TYPE_FIRST_PASS	=00000011	129	(1)							
DS$K_TYPE_GENERAL	=00000000	129	(1)							
DS$K_TYPE_GENERAL_ERROR	=00000003	129	(1)							
DS$K_TYPE_NO_TESTS	=00000012	129	(1)							
DS$K_TYPE_PARAM_ERROR	=0000001C	129	(1)							

```
DS$K_TYPE_PROGRAM_END           =00000010      129     (1)     #-440     (2)
DS$K_TYPE_PROGRAM_INFO          =00000017      129     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      129     (1)
DS$K_TYPE_QIO_INVADP            =00000024      129     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      129     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      129     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      129     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      129     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      129     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      129     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      129     (1)     #-274     (1)     #-465     (2)
DS$K_TYPE_START_ERR             =00000018      129     (1)
DS$K_TYPE_START_LIST            =00000025      129     (1)
DS$K_TYPE_SUMMARY               =0000000E      129     (1)
DS$K_TYPE_USER_PROMPT           =00000C02      129     (1)
DS$K_WARNING                    =00000000      123     (1)
DS$M_ABRTFLG                    =00000040      126     (1)
DS$M_BADTIME                    =00100000      126     (1)
DS$M_BATCH                      =00400000      126     (1)
DS$M_BRKCLR                     =00001000      126     (1)
DS$M_BRKPT                      =00000800      126     (1)
DS$M_CHARFLG                    =00000100      126     (1)
DS$M_CMDFLG                     =00000080      126     (1)
DS$M_CTRLC                      =00000001      126     (1)
DS$M_CTRLO                      =00010000      126     (1)
DS$M_DEVFLG                     =00000200      126     (1)
DS$M_DISABLCC                   =01000000      126     (1)
DS$M_DONFLG                     =00002000      126     (1)
DS$M_ERRFLG                     =00000010      126     (1)
DS$M_EXCEPT                     =00080000      126     (1)
DS$M_EXETST                     =00040000      126     (1)
DS$M_HLTFLG                     =00000008      126     (1)
DS$M_LODFLG                     =00000002      126     (1)
DS$M_MEMMGT                     =00008000      126     (1)
DS$M_OUTPUT                     =00800000      126     (1)
DS$M_RUBFLG                     =00000020      126     (1)
DS$M_SCRIPT                     =00200000      126     (1)
DS$M_SETIMR                     =02000000      126     (1)
DS$M_STRFLG                     =00000004      126     (1)
DS$M_SUBT                       =00004000      126     (1)
DS$M_SYSFLG                     =00000400      126     (1)
DS$M_TIMRON                     =00020000      126     (1)
DS$SUMMARY                       00000000-XR                    443     (2)
DS$V_ABRTFLG                    =00000006      126     (1)
DS$V_BADTIME                    =00000014      126     (1)
DS$V_BATCH                      =00000016      126     (1)
DS$V_BRKCLR                     =0000000C      126     (1)
DS$V_BRKPT                      =0000000B      126     (1)
DS$V_CHARFLG                    =00000008      126     (1)
DS$V_CMDFLG                     =00000007      126     (1)     #-277     (1)     #-422     (2)     #-468     (2)
DS$V_CTRLC                      =00000000      126     (1)
DS$V_CTRLO                      =00000010      126     (1)     #-424     (2)     #-446     (2)
DS$V_DEVFLG                     =00000009      126     (1)
DS$V_DISABLCC                   =00000018      126     (1)
DS$V_DONFLG                     =0000000D      126     (1)
DS$V_ERRFLG                     =00000004      126     (1)     #-186     (1)     #-246     (1)     #-370     (2)     #-536     (2)
                                                               #-583     (3)
```

```
DS$V_EXCEPT                     =00000013       126     (1)
DS$V_EXETST                     =00000012       126     (1)
DS$V_HLTFLG                     =00000003       126     (1)
DS$V_LODFLG                     =00000001       126     (1)
DS$V_MEMMGT                     =0000000F       126     (1)
DS$V_OUTPUT                     =00000017       126     (1)
DS$V_RUBFLG                     =00000005       126     (1)
DS$V_SCRIPT                     =00000015       126     (1)
DS$V_SETIMR                     =00000019       126     (1)
DS$V_STRFLG                     =00000002       126     (1)
DS$V_SUBT                       =0000000E       126     (1)     #-243     (1)     #-366     (2)     #-410     (2)
DS$V_SYSFLG                     =0000000A       126     (1)
DS$V_TIMRON                     =00000011       126     (1)
DS$_ARITH                       =006600D0       123     (1)
DS$_ASBE                        =00660118       123     (1)
DS$_BADLINK                     =006600F0       123     (1)
DS$_BADTYPE                     =006600E8       123     (1)
DS$_BIIC                        =00660120       123     (1)
DS$_CHME                        =006600A8       123     (1)
DS$_CHMK                        =006600E0       123     (1)
DS$_DEVNAME                     =00660108       123     (1)
DS$_ERROR                       =00660002       123     (1)     #-581     (3)
DS$_FHWE                        =00660068       123     (1)
DS$_FRAGBUF                     =00660080       123     (1)
DS$_ICBUSY                      =006600C8       123     (1)
DS$_ICERR                       =006600C0       123     (1)
DS$_IHWE                        =00660060       123     (1)
DS$_ILLCHAR                     =00660018       123     (1)
DS$_ILLPAGCNT                   =00660078       123     (1)
DS$_ILLUNIT                     =00660100       123     (1)
DS$_INSFMEM                     =00660050       123     (1)
DS$_IPL2HI                      =006600B8       123     (1)
DS$_IVADDR                      =00660040       123     (1)
DS$_IVVECT                      =00660038       123     (1)
DS$_KRNLSTK                     =00660090       123     (1)
DS$_LOGIC                       =00660070       123     (1)
DS$_MCHK                        =00660088       123     (1)
DS$_MMOFF                       =00660058       123     (1)
DS$_NEEDUNIT                    =006600F8       123     (1)
DS$_NODE                        =00660128       123     (1)
DS$_NOPCS                       =00660110       123     (1)
DS$_NORMAL                      =00660001       123     (1)     #-586     (3)
DS$_NOSUPPORT                   =006600B1       123     (1)
DS$_NOTDON                      =00660030       123     (1)
DS$_NOTIMP                      =006600B0       123     (1)       123     (1)
DS$_NULLSTR                     =00660010       123     (1)
DS$_OVERFLOW                    =00660008       123     (1)
DS$_POWER                       =00660098       123     (1)
DS$_PROGERR                     =00660020       123     (1)
DS$_SEVERE                      =00660004       123     (1)
DS$_TRANSL                      =006600A0       123     (1)
DS$_TRUNCATF                    =00660028       123     (1)
DS$_UNEXPINT                    =006600D8       123     (1)
DS$_VASFULL                     =00660048       123     (1)
DS$_WARNING                     =00660000       123     (1)       123     (1)
DSA$GL_FLAGS                     0000FE00                         276     (1)       369     (2)       427     (2)       450     (2)
                                                                  467     (2)       535     (2)       585     (3)
```

```
DSA$GL_MSGTYP                   0000FE40                        #-449   (2)
DSA$GL_PASSES                   0000FE08                        #-389   (2)     #-396   (2)
DSA$GL_PASSNO                   0000FE54                          388   (2)     #-440   (2)
DSA$GL_SUBTNO                   0000FE4C                        #-274   (1)     #-292   (1)     #-356   (2)     #-440   (2)
                                                                #-465   (2)     #-485   (2)
DSA$GL_TESTNO                   0000FE50                        #-259   (1)     #-274   (1)     #-285   (1)     #-354   (2)
                                                                #-440   (2)     #-465   (2)     #-484   (2)
DSA$V_HALT                      =00000000                       #-276   (1)     #-467   (2)
DSA$V_LOOP                      =000C0002                       #-368   (2)     #-534   (2)     #-584   (3)
DSA$V_QA                        =0000000F                       #-427   (2)     #-450   (2)
DSQA$K_DISPAT_AFTER_INIT        =00000014       127     (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013       127     (1)
DSQA$K_DISPAT_CALLTEST          =00000015       127     (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018       127     (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000C01       127     (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002       127     (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000       127     (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003       127     (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004       127     (1)
DSQA$K_DUMMY_T                  =00000007       127     (1)
DSQA$K_ERROR_ERROR_BGN          =00000006       127     (1)
DSQA$K_ERROR_ERROR_END          =00000005       127     (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008       127     (1)
DSQA$K_LOOP_DSX$BGNSUB          =00000009       127     (1)     #-241   (1)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B       127     (1)     #-531   (2)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A       127     (1)     #-345   (2)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E       127     (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C       127     (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F       127     (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010       127     (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D       127     (1)
DSQA$K_QA_DSX$BRANCH            =00000011       127     (1)
DSQA$K_START_BEFORE_HEADER      =00000017       127     (1)
DSQA$K_START_VRRESTART          =00000012       127     (1)
DSQA$K_START_VRSTART            =00000016       127     (1)
DSX$BGNSUB                       00000015-R     239     (1)
DSX$CKLOOP                       00000224-R     529     (2)
DSX$ENDSUB                       000000B3-R     343     (2)
DSX$ESCAPE                       00000000-R     183     (1)
DSX$INLOOP                       00000264-R     578     (3)
DSX$PRINT                        00000000-XR                     274     (1)     440     (2)     465     (2)
DS_CLEANUP                       00000000-XR                     444     (2)
DS_ERRSUP                        00000000-XR                     279     (1)     470     (2)
FALSE                           =00000000       128     (1)
FMTLOOPDON                       0000004F-R     142     (1)     440     (2)
FMTSEQERR                        00000005-R     139     (1)     274     (1)     465     (2)
INIT_CONTEXT                     00000000-XR                     442     (2)
KB_CHECK                         00000000-XR                     185     (1)     293     (1)     346     (2)     533     (2)
                                                                580     (3)
OFF                             =00000000       128     (1)
ON                              =00000001       128     (1)
QA$AOB_CHECK_STATE               00000000-XR                    429     (2)
QA$K_ABORT_NOW                  =00000007       128     (1)
QA$K_APT_PHASE_ONE              =0000000B       128     (1)
QA$K_APT_PHASE_TWO              =0000000C       128     (1)
QA$K_BAD_ADDRESS                =00000009       128     (1)
QA$K_CONTROL_C_CONT             =0000000A       128     (1)
```

QA$K_DEALLOCATION	=0000000E	128	(1)							
QA$K_ERROR_PHASE_ONE	=00000005	128	(1)							
QA$K_ERROR_PHASE_THREE	=00000007	128	(1)							
QA$K_ERROR_PHASE_TWO	=00000006	128	(1)							
QA$K_FAILURE	=00000000	128	(1)							
QA$K_FIRST_BRANCH_CODE	=00000000	128	(1)							
QA$K_FIRST_TIME	=00000005	128	(1)							
QA$K_INIT_CONTEXT_DONE	=00000003	128	(1)							
QA$K_LAST_BRANCH_CODE	=00000025	128	(1)							
QA$K_LOOP_ON_SUBTEST	=00000003	128	(1)	#-428	(2)					
QA$K_LOOP_ON_TEST	=00000002	128	(1)							
QA$K_LOOP_TEST_DONE	=00000004	128	(1)							
QA$K_MEMORY_MANAGE	=0000000D	128	(1)							
QA$K_MULTIPLE_PASS	=00000001	128	(1)							
QA$K_NODEF	=00000C00	128	(1)							
QA$K_NORMAL_START	=00000000	128	(1)							
QA$K_NO_HEADER	=00000006	128	(1)							
QA$K_NUMBER_OF_CHECKS	=0000000F	128	(1)							
QA$K_NUMBER_QA_FLAGS	=00000008	128	(1)							
QA$K_NUMB_ROUTINE_STATES	=00000004	128	(1)							
QA$K_QA_DEBUG	=00000001	128	(1)							
QA$K_QA_DONE	=00000002	128	(1)							
QA$K_RUN_BACKWARDS	=00000004	128	(1)							
QA$K_SECTION	=00000008	128	(1)							
QA$K_SUCCESS	=00000001	128	(1)							
QA$MAIN	00000000-XR			241	(1)	345	(2)	531	(2)	
QA$V_CHECK_DONE	=00000003	128	(1)							
QA$V_DOING_CLEANUP	=00000002	128	(1)							
QA$V_ERROR_FOUND	=00000001	128	(1)							
QA$V_INIT_DONE	=00000000	128	(1)							
RBGNSUBX	000000B2-R	295	(1)							
RCKLOOPX	00000263-R	546	(2)	#-535	(2)	#-536	(2)	#-541	(2)	
RESCAPEX	00000014-R	189	(1)	#-186	(1)					
SIZ...	=00000001	126	(1)	126	(1)					
TRUE	=00000001	128	(1)							
VRRESTART	00000000-XR			#-451	(2)					

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_PRINT_S	1	274 (1)	274 (1) 440 (2) 465 (2)
$DEF	1	129 (1)	
$DEFINI	1	122 (1)	122 (1) 124 (1)
$DS_DSADEF	5	122 (1)	122 (1)
$DS_DSDEF	2	123 (1)	123 (1)
$DS_HDRDEF	2	124 (1)	124 (1)
$DS_QADEF	2	128 (1)	128 (1)
$DS_SUMMARY_S	1	443 (2)	443 (2)
$DS_TYPEDEF	4	129 (1)	129 (1)
$EQU	1	129 (1)	123 (1) 127 (1) 128 (1) 129 (1)
$EQULS1	1	129 (1)	123 (1) 127 (1) 128 (1) 129 (1)
$EQULST	1	123 (1)	123 (1) 127 (1) 128 (1) 129 (1)
$GBLINI	2		123 (1) 126 (1) 127 (1) 128 (1)
			129 (1)
$PRINT	2	269 (1)	269 (1) 433 (2) 460 (2)
$PUSHADR	1	279 (1)	279 (1) 470 (2)
$VIELD	1		126 (1)
$VIELD1	1	129 (1)	126 (1)
APTDEF	1	125 (1)	125 (1)
BR_IF_ERRFLG	1	370 (2)	370 (2)
BR_IF_NOT_ERRFLG	1	186 (1)	186 (1) 536 (2) 583 (3)
BR_IF_NOT_QA	1	427 (2)	427 (2) 450 (2)
CLEAR_CTRLO	1	446 (2)	446 (2)
CLEAR_ERRFLG	1	246 (1)	246 (1)
DSFDEF	3	126 (1)	126 (1)
DSQA	2	127 (1)	127 (1)
DS_QADEFS	6	128 (1)	128 (1)
ERRSUP_S	1	279 (1)	279 (1) 470 (2)
MODNAM	1	137 (1)	137 (1)
QA_MAIN	1	241 (1)	241 (1) 345 (2) 531 (2)
SET_HALT	1	276 (1)	276 (1) 467 (2)
SET_SUBT	1	410 (2)	410 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.26
Command processing	140	00:00:00.77	00:00:01.84
Pass 1	700	00:00:11.60	00:00:25.96
Symbol table sort	3	00:00:00.50	00:00:00.69
Pass 2	163	00:00:02.25	00:00:06.28
Symbol table output	41	00:00:00.24	00:00:00.68
Psect synopsis output	8	00:00:00.04	00:00:00.04
Cross-reference output	68	00:00:01.29	00:00:03.28
Assembler run totals	1161	00:00:16.83	00:00:39.04

The working set limit was 1000 pages.

58268 bytes (114 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 385 non-local and 22 local symbols.
591 source lines were read in Pass 1, producing 0 object records in Pass 2.
99 pages of virtual memory were used to define 31 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	7
DRB1:[DS.WORK]DS.MLB;218	15
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	28

577 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) LOOP/UPDA=(LOOP.UPD,LOOP.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DI
```

Table of contents

(1) 63 MASSBUS ADAPTER INTERRUPT DISPATCHER
(1) 149 MASSBUS ADAPTER INITIALIZATION

```
                        0000     1          .TITLE  MBAINT - MASSBUS ADAPTER INTERRUPT DISPATCHER
                        0000     2          .IDENT  /01/
                    00000000     3          .PSECT  SEP, SHR, WRT, EXE, LONG
                        0000     4
                        0000     5  ;
                        0000     6  ;*****************************************************************************
                        0000     7  ;*                                                                           *
                        0000     8  ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000     9  ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
                        0000    10  ;*                                                                           *
                        0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED    *
                        0000    12  ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
                        0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
                        0000    14  ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
                        0000    16  ;*  TRANSFERRED.                                                             *
                        0000    17  ;*                                                                           *
                        0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
                        0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                        0000    20  ;*  CORPORATION.                                                             *
                        0000    21  ;*                                                                           *
                        0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS    *
                        0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000    24  ;*                                                                           *
                        0000    25  ;*****************************************************************************
                        0000    26  ;
                        0000    27  ; D. N. CUTLER 30-JAN-77
                        0000    28  ;
                        0000    29  ; MASSBUS ADAPTER INTERRUPT DISPATCHER
                        0000    30  ; R. S. RIGGS  14-Jun-1980 Minor mods to accomodate SUPERVISOR
                        0000    31  ;
                        0000    32  ; MASSBUS ADAPTER INTERRUPT DISPATCHER
                        0000    33  ;
                        0000    34  ; MODIFIED BY:
                        0000    35  ;
                        0000    36  ;       V03     RLR0001         Robert L. Rappaport     15-NOV-1979
                        0000    37  ;               Changed attention summary bit clearing logic to allow
                        0000    38  ;               support of TM78 Tape Controller.  Specifically, for
                        0000    39  ;               all MASSBUS multi-device controllers (currently TM03
                        0000    40  ;               and TM78), the responsibility for clearing the attention
                        0000    41  ;               summary bit for the device has been removed from this
                        0000    42  ;               module and placed in the individual Device Drivers.
                        0000    43  ;               For MASSBUS single device controllers (DISKS) the
                        0000    44  ;               logic remains as it was.  That is the attention summary
                        0000    45  ;               bit is cleared here before dispatching to the specific Driver.
                        0000    46  ;
                        0000    47  ;
                        0000    48  ;       V02     NPK0001         N. KRONENBERG           4-NOV-1979
                        0000    49  ;               ADDED CHECK FOR CBHUNG ON MBA INTERRUPT
                        0000    50  ;
```

```
                        0000     52 ;
                        0000     53 ; MACRO LIBRARY CALLS
                        0000     54 ;
                        0000     55
                        0000     56          $ADPDEF                         ;DEFINE ADP OFFSETS
                        0000     57          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     58          $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000     59          $MBADEF                         ;DEFINE MBA REGISTER OFFSETS
                        0000     60          $IDBDEF                         ;DEFINE IDB OFFSETS
                        0000     61          $UCBDEF                         ;DEFINE UCB OFFSETS
```

```
                                63          .SBTTL  MASSBUS ADAPTER INTERRUPT DISPATCHER
                                64 ;+
                                65 ; MBA$INT - MASSBUS ADAPTER INTERRUPT DISPATCHER
                                66 ;
                                67 ; THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN INTERRUPT OCCURS
                                68 ; ON A MASSBUS ADAPTER. THE STATE OF THE STACK ON ENTRY IS:
                                69 ;
                                70 ;       00(SP) = ADDRESS OF IDB ADDRESS.
                                71 ;       04(SP) = SAVED R2.
                                72 ;       08(SP) = SAVED R3.
                                73 ;       12(SP) = SAVED R4.
                                74 ;       16(SP) = SAVED R5.
                                75 ;       20(SP) = INTERRUPT PC.
                                76 ;       24(SP) = INTERRUPT PSL.
                                77 ;
                                78 ; INTERRUPT DISPATCHING OCCURS AS FOLLOWS:
                                79 ;
                                80 ;       IF THE INTERRUPTING ADAPTER IS CURRENTLY OWNED AND THE OWNER UNIT
                                81 ;       IS EXPECTING AN INTERRUPT, THEN THAT UNIT IS DISPATCHED FIRST. ALL
                                82 ;       OTHER UNITS ARE DISPATCHED BY READING THE ATTENTION SUMMARY REG-
                                83 ;       ISTER AND SCANNING FOR UNITS THAT HAVE ATTENTION SET. AS EACH UNIT
                                84 ;       IS FOUND, ITS ATTENTION SUMMARY BIT IS CLEARED AND THEN A TEST IS
                                85 ;       MADE TO DETERMINE IF AN INTERRUPT IS EXPECTED ON THE UNIT. IF YES,
                                86 ;       THEN THE DRIVER IS CALLED AT ITS INTERRUPT RETURN ADDRESS. ELSE
                                87 ;       THE DRIVER IS CALLED AT ITS UNSOLICITED INTERRUPT ADDRESS. AS EACH
                                88 ;       CALL TO THE DRIVER RETURNS, THE ATTENTION SUMMARY REGISTER IS RE-
                                89 ;       READ AND AN ATTEMPT IS MADE TO FIND ANOTHER UNIT TO DISPATCH. WHEN
                                90 ;       NO UNITS REQUESTING ATTENTION REMAIN, THE INTERRUPT IS DISMISSED.
                                91 ;-
                                92
                                93 MBA$INT::                                    ;MASSBUS ADAPTER INTERRUPT DISPATCHER
          53    00 BE    D0     94          MOVL    @(SP),R3                    ;GET ADDRESS OF IDB
                54 63    D0     95          MOVL    IDB$L_CSR(R3),R4            ;GET ADDRESS OF CONFIGURATION STATUS REGISTE
08 A4   00800000 8F    D3     96          BITL    #MBA$M_SR_CBHUNG,MBA$L_SR(R4)   ;CHECK FOR MBA HUNG
                   6D  12     97          BNEQ    50$                             ;BRANCH IF HUNG
          55    04 A3    D0     98          MOVL    IDB$L_OWNER(R3),R5          ;GET OWNER UNIT UCB ADDRESS
                   09  13     99          BEQL    10$                             ;IF EQL NO OWNER
          52    74 A5    9A    100          MOVZBL  UCB$B_SLAVE(R5),R2          ;GET OWNER SLAVE CONTROLLER NUMBER
       21 58 A5    01    E0    101          BBS     #UCB$V_INT,UCB$W_STS(R5),20$ ;IF SET, INTERRUPT EXPECTED
          53    00 BE    D0    102 10$:     MOVL    @(SP),R3                    ;RETRIEVE ADDRESS OF IDB
                54 63    D0    103          MOVL    IDB$L_CSR(R3),R4            ;RETRIEVE MBA CONFIGURATION REGISTER ADDRESS
             08 A4 00    D2    104          MCOML   #0,MBA$L_SR(R4)             ;CLEAR ALL MBA STATUS BITS
          52  0410 C4    D0    105          MOVL    MBA$L_AS(R4),R2             ;READ ATTENTION SUMMARY REGISTER
52        52    08 00    EA    106          FFS     #0,#8,R2,R2                 ;FIND FIRST UNIT REQUESTING ATTENTION
                   0A  12     107          BNEQ    20$                         ;IF NEQ UNIT FOUND
                5E 04 C0     108          ADDL    #4,SP                       ;REMOVE IDB ADDRESS FROM STACK
                52 8E 7D     109          MOVQ    (SP)+,R2                    ;RESTORE REGISTERS
                54 8E 7D     110          MOVQ    (SP)+,R4                    ;
                   02  0040  111          REI                                 ;
          55    14 A342  D0    112 20$:     MOVL    IDB$L_UCBLST(R3)[R2],R5     ;GET ADDRESS OF UCB OR INTERRUPT DISPATCHER
                2F 55    E8    113          BLBS    R5,40$                      ;IF LBS INTERRUPT DISPATCHER FOR MULTI-
                         0049  114                                              ; DEVICE CONTROLLER
0410 C4    01    52    78    115          ASHL    R2,#1,MBA$L_AS(R4)          ;CLEAR ATTENTION SUMMARY BIT
                55    D5    116          TSTL    R5                          ;SEE IF UCB DEFINED
                   CD  13     117          BEQL    10$                         ;IF EQL NONE DEFINED
       09 58 A5    01    E5    118          BBCC    #UCB$V_INT,UCB$W_STS(R5),30$ ;IF CLR, INTERRUPT NOT EXPECTED
          53    10 A5    7D    119          MOVQ    UCB$L_FR3(R5),R3            ;RESTORE DRIVER CONTEXT
```

```
                 0C B5   16  005C   120          JSB     @UCB$L_FPC(R5)          ;CALL DRIVER AT INTERRUPT RETURN ADDRESS
                    BF   11  005F   121          BRB     10$                     ;
           53    24 A5   D0  0061   122 30$:     MOVL    UCB$L_DDB(R5),R3        ;GET ADDRESS OF DDB
           53    0C A3   D0  0065   123          MOVL    DDB$L_DDT(R3),R3        ;GET ADDRESS OF DDT
           52    04 A3   D0  0069   124          MOVL    DDT$L_UNSOLINT(R3),R2   ;CALCULATE ADDRESS OF UNSOLICITED INTERRUPT
        03 52      10   E0  006D   125          BBS     #16,R2,35$              ;++++ BRANCH IF SUPERVISOR ADDR
              52   53   C0  0071   126          ADDL    R3,R2                   ;++++ MAKE ABSOLUTE ADDRESS
                   62   16  0074   127 35$:     JSB     (R2)                    ;CALL UNSOLICITED INTERRUPT ROUTINE
                   A8   11  0076   128          BRB     10$                     ;
                   7E   DC  0078   129 40$:     MOVPSL  -(SP)                   ;READ CURRENT PSL
                   75   16  007A   130          JSB     -(R5)                   ;CALL SLAVE CONTROLLER INTERRUPT DISPATCHER
                   A2   11  007C   131          BRB     10$                     ;
                            007E   132
                            007E   133 ;
                            C07E   134 ; IN CASE OF CBHUNG SAVE MBA INFORMATIION FOR BUGCHECK LOG AND
                            007E   135 ; BUGCHECK.  CBHUNG IS IMPLEMENTED ONLY ON THE VAX 11/750 CPU.
                            007E   136 ; IT MEANS THAT AN ACCESS TO A REGISTER OF AN EXISTENT CONTROLLER
                            007E   137 ; FAILED TO COMPLETE IN 1.5 USEC.
                            007E   138 ;
                            007E   139
           55    04 A3   D0  007E   140 50$:     MOVL    IDB$L_OWNER(R3),R5          ;SAVE OWNER UCB IF ANY
           50    08 A4   D0  0082   141          MOVL    MBA$L_SR(R4),R0             ;SAVE MBA STATUS REGISTER,
              51    64   D0  0086   142          MOVL    MBA$L_CSR(R4),R1            ; CONFIGURATION REGISTER,
           52    14 A4   D0  0089   143          MOVL    MBA$L_DR(R4),R2             ; DIAGNOSTIC REGISTER,
           53    10 A3   D0  008D   144          MOVL    IDB$L_ADP(R3),R3            ;GET ADP ADDRESS
           53    0C A3   3C  0091   145          MOVZWL  ADP$W_TR(R3),R3             ;SAVE NEXUS NUMBER TO IDENTIFY
                            0095   146                                              ; OFFENDING MBA
                            0095   147          BUG_CHECK MBACBHUNG,FATAL           ;FATAL ERROR
```

```
                        00AB   149          .SBTTL  MASSBUS ADAPTER INITIALIZATION
                        00AB   150 ;+
                        00AB   151 ; MBA$INITIAL - MASSBUS ADAPTER INITIALIZATION
                        00AB   152 ;
                        00AB   153 ; THIS ROUTINE IS CALLED VIA A JSB INSTRUCTION AT SYSTEM STARTUP AND AFTER
                        00AB   154 ; A POWER RECOVERY RESTART TO ALLOW INITIALIZATION OF MASSBUS ADAPTERS.
                        00AB   155 ;
                        00AB   156 ; INPUTS:
                        00AB   157 ;
                        00AB   158 ;       R4 = CSR ADDRESS OF MASSBUS ADAPTER.
                        00AB   159 ;       R5 = ADDRESS OF ADAPTER IDB.
                        00AB   160 ;
                        00AB   161 ;       ALL INTERRUPTS ARE LOCKED OUT.
                        00AB   162 ;
                        00AB   163 ; OUTPUTS:
                        00AB   164 ;
                        00AB   165 ;       THE MASSBUS ADAPTER IS INITIALIZED AND INTERRUPTS ARE ENABLED.
                        00AB   166 ;-
                        00AB   167
                        00AB   168 MBA$INITIAL::                            ;MASSBUS ADAPTER INITIALIZATION
04 A4   01   D0         00AB   169          MOVL    #MBA$M_CR_INIT,MBA$L_CR(R4) ;INITIALIZE MASSBUS ADAPTER
04 A4   04   D0         00AF   170          MOVL    #MBA$M_CR_IE,MBA$L_CR(R4) ;ENABLE INTERRUPTS
                05      00B3   171          RSB                               ;
                        00B4   172
                        00B4   173          .END
```

ADP$B_NUMBER	00000008	D		IDB$W_UNITS	0000000C	D		UCB$L_DUETIM	0000005C	D
ADP$B_PORT	00000020	D		MBA$INITIAL	000000AB	RG	D	01	UCB$L_DX_BFPNT	0000009C	D
ADP$B_TYPE	0000000A	D		MBA$INT	000000C0	RG	D	01	UCB$L_DX_BUF	00000098	D
ADP$C_DRADPLEN	00000014	D		MBA$L_AS	= 00000410		D		UCB$L_DX_RXDB	000000A0	D
ADP$C_MBAADPLEN	00000014	D		MBA$L_CR	= 00000004		D		UCB$L_EMB	00000078	D
ADP$C_MPMADPLEN	00000070	D		MBA$L_CSR	= 00000000		D		UCB$L_FIRST	00000014	D
ADP$C_UBAADPLEN	00000070	D		MBA$L_DR	= 00000014		D		UCB$L_FPC	0000000C	D
ADP$K_DRADPLEN	00000014	D		MBA$L_SR	= 00000008		D		UCB$L_FQBL	00000004	D
ADP$K_MBAADPLEN	00000014	D		MBA$M_CR_IE	= 00000004		D		UCB$L_FQFL	00000000	D
ADP$K_MPMADPLEN	00000070	D		MBA$M_CR_INIT	= 00000001		D		UCB$L_FR3	00000010	D
ADP$K_UBAADPLEN	00000070	D		MBA$M_SR_CBHUNG	= 00800000		D		UCB$L_FR4	00000014	D
ADP$L_CRB	00000010	D		SIZ...	= 00000002		D		UCB$L_IOQBL	00000044	D
ADP$L_CSR	00000000	D		UCB$B_AMOD	00000053		D		UCB$L_IOQFL	00000040	D
ADP$L_DPQBL	00000018	D		UCB$B_CEX	00000077		D		UCB$L_IRP	0000004C	D
ADP$L_DPQFL	00000014	D		UCB$B_CM1	0000004A		D		UCB$L_LINK	0000002C	D
ADP$L_INTD	00000064	D		UCB$B_CM2	0000004B		D		UCB$L_LOGADR	00000064	D
ADP$L_LINK	00000004	D		UCB$B_DEVCLASS	00000038		D		UCB$L_MAXBLOCK	00000084	D
ADP$L_MRQBL	00000020	D		UCB$B_DEVTYPE	00000039		D		UCB$L_MB_MBX	0000007C	D
ADP$L_MRQFL	0000001C	D		UCB$B_DIPL	00000052		D		UCB$L_MB_PORT	0000008C	D
ADP$L_PRQQBL	00000018	D		UCB$B_DX_SCTCNT	000000A6		D		UCB$L_MB_RAST	00000078	D
ADP$L_PRQQFL	00000014	D		UCB$B_ERTCNT	00000070		D		UCB$L_MB_SHB	00000080	D
ADP$L_SHB	0000001C	D		UCB$B_ERTMAX	00000071		D		UCB$L_MB_WAST	00000074	D
ADP$L_VECTOR	00000010	D		UCB$B_FEX	00000076		D		UCB$L_MB_WIOQBL	00000088	D
ADP$W_ADPTYPE	0000000E	D		UCB$B_FIPL	0000000B		D		UCB$L_MB_WIOQFL	00000084	D
ADP$W_DPBITMAP	00000024	D		UCB$B_LOCSRV	0000003C		D		UCB$L_MEDIA	0000008C	D
ADP$W_MRBITMAP	00000026	D		UCB$B_OFFNDX	00000094		D		UCB$L_NT_DATSSB	00000074	D
ADP$W_SIZE	00000008	D		UCB$B_OFFRTC	00000095		D		UCB$L_NT_INTSSB	00000078	D
ADP$W_TR	0000000C	D		UCB$B_REMSRV	0000003D		D		UCB$L_OPCNT	00000060	D
BUG$CHECK	********	X	01	UCB$B_SECTORS	0000003C		D		UCB$L_OWNUIC	0000001C	D
DDB$B_ACPCLASS	00000013	D		UCB$B_SLAVE	00000074		D		UCB$L_PID	00000028	D
DDB$B_TYPE	0000000A	D		UCB$B_SPR	00000075		D		UCB$L_RQBL	00000004	D
DDB$C_LENGTH	00000034	D		UCB$B_STATE	00000052		D		UCB$L_RQFL	00000000	D
DDB$K_LENGTH	00000034	D		UCB$B_TRACKS	0000003D		D		UCB$L_SVAPTE	00000068	D
DDB$L_ACPD	00000010	D		UCB$B_TT_CRFILL	0000009D		D		UCB$L_SVPN	00000064	D
DDB$L_DDT	0000000C	D		UCB$B_TT_DECRF	000000A1		D		UCB$L_TT_DECHAR	000000A8	D
DDB$L_LINK	00000000	D		UCB$B_TT_DELFF	000000A2		D		UCB$L_TT_RDUE	0000008C	D
DDB$L_UCB	00000004	D		UCB$B_TT_DESPCE	000000A0		D		UCB$L_TT_RTIMOU	000000B8	D
DDB$T_DRVNAME	00000024	D		UCB$B_TT_DETYPE	000000A4		D		UCB$L_VCB	00000030	D
DDB$T_NAME	00000014	D		UCB$B_TT_LFFILL	0000009E		D		UCB$T_PARTNER	0000000C	D
DDB$W_SIZE	00000008	D		UCB$B_TT_SPEED	0000009C		D		UCB$V_INT	= 00000001	D
DDT$L_ALTSTART	0000001C	D		UCB$B_TYPE	0000000A		D		UCB$W_BCNT	0000006E	D
DDT$L_CANCEL	0000000C	D		UCB$B_VERTSZ	0000003F		D		UCB$W_BCR	00000096	D
DDT$L_FDT	00000008	D		UCB$C_LENGTH	00000074		D		UCB$W_BOFF	0000006C	D
DDT$L_REGDUMP	00000010	D		UCB$C_MB_LENGTH	00000090		D		UCB$W_BUFQUO	00000018	D
DDT$L_START	00000000	D		UCB$C_TT_LENGTH	000000BC		D		UCB$W_BYTES[illegible]O	0000003E	D
DDT$L_UNITINIT	00000018	D		UCB$K_LENGTH	00000074		D		UCB$W_CHARGE	0000004A	D
DDT$L_UNSOLINT	00000004	D		UCB$K_MB_LENGTH	00000090		D		UCB$W_CYLINDER[illegible]	0000003E	D
DDT$W_DIAGBUF	00000014	D		UCB$K_TT_LENGTH	000000BC		D		UCB$W_DA	0000008C	D
DDT$W_ERRORBUF	00000016	D		UCB$L_AMB	00000054		D		UCB$W_DC	0000008E	D
IDB$B_TYPE	0000000A	D		UCB$L_ASTQBL	00000010		D		UCB$W_DEVBU[illegible]	0000003A	D
IDB$C_LENGTH	00000034	D		UCB$L_ASTQFL	0000000C		D		UCB$W_DEVST[illegible]	0000005A	D
IDB$K_LENGTH	00000034	D		UCB$L_CPID	0000005C		D		UCB$W_DIRSEQ	00000088	D
IDB$L_ADP	00000010	D		UCB$L_CRB	00000020		D		UCB$W_DSTADDR	00000018	D
IDB$L_CSR	00000000	D		UCB$L_DDB	00000024		D		UCB$W_DX_BCR	000000A4	D
IDB$L_OWNER	00000004	D		UCB$L_DEVCHAR	00000C34		D		UCB$W_ECT	00000090	D
IDB$L_UCBLST	00000014	D		UCB$L_DEVDEPEND	0000003C		D		UCB$W_EC2	00000092	D
IDB$W_SIZE	00000008	D		UCB$L_DPC	00000080		D		UCB$W_ERRCNT	00000072	D

UCB$W_FUNC	0000007E	D
UCB$W_MB_SEED	00000000	D
UCB$W_MSGCNT	00000016	D
UCB$W_MSGMAX	00000014	D
UCB$W_NT_CHAN	0000007C	D
UCB$W_OFFSET	0000008A	D
UCB$W_REFC	00000050	D
UCB$W_SIZE	00000008	D
UCB$W_SRCADDR	0000001A	D
UCB$W_STS	00000058	D
UCB$W_TT_DESIZE	000000A5	D
UCB$W_UNIT	00000048	D
UCB$W_VPROT	0000001A	D

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
SEP	000000B4 (180.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	000000BC (188.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...					
ADP$W_TR	0000000C			#-145	(1)				
BUG$CHECK	00000000-XR			147	(1)				
DDB$L_DDT	0000000C			#-123	(1)				
DDT$L_UNSOLINT	00000004			#-124	(1)				
IDB$L_ADP	00000010			#-144	(1)				
IDB$L_CSR	00000000			#-103	(1)	#-95	(1)		
IDB$L_OWNER	00000004			#-140	(1)	#-98	(1)		
IDB$L_UCBLST	00000014			#-112	(1)				
MBA$INITIAL	000000AB-R	168	(1)						
MBA$INT	00000000-R	93	(1)						
MBA$L_AS	=00000410			#-105	(1)	#-115	(1)		
MBA$L_CR	=00000004			#-169	(1)	#-170	(1)		
MBA$L_CSR	=00000000			#-142	(1)				
MBA$L_DR	=00000014			#-143	(1)				
MBA$L_SR	=00000008			#-104	(1)	#-141	(1)	#-96	(1)
MBA$M_CR_IE	=00000004			#-170	(1)				
MBA$M_CR_INIT	=00000001			#-169	(1)				
MBA$M_SR_CBHUNG	=00800000			#-96	(1)				
UCB$B_SLAVE	00000074			#-100	(1)				
UCB$L_DDB	00000024			#-122	(1)				
UCB$L_FPC	0000000C			120	(1)				
UCB$L_FR3	00000010			#-119	(1)				
UCB$V_INT	=00000001			#-101	(1)	#-118	(1)		
UCB$W_STS	00000058			101	(1)	118	(1)		

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ADPDEF	2	56 (1)	56 (1)
$DDBDEF	1	57 (1)	57 (1)
$DDTDEF	1	58 (1)	58 (1)
$DEFINI	1	56 (1)	56 (1) 57 (1) 58 (1) 59 (1) 60 (1) 61 (1)
$IDBDEF	1	60 (1)	60 (1)
$MBADEF	5	59 (1)	59 (1)
$UCBDEF	10	61 (1)	61 (1)
BUG_CHECK	1	147 (1)	147 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.10	00:00:00.33
Command processing	139	00:00:00.81	00:00:01.61
Pass 1	326	00:00:06.25	00:00:07.69
Symbol table sort	0	00:00:00.42	00:00:00.43
Pass 2	62	00:00:00.90	00:00:01.22
Symbol table output	8	00:00:00.11	00:00:00.15
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	13	00:00:00.13	00:00:00.14
Assembler run totals	590	00:00:08.76	00:00:11.60

The working set limit was 1000 pages.
25474 bytes (50 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 328 non-local and 6 local symbols.
173 source lines were read in Pass 1, producing 0 object records in Pass 2.
41 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	5
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	12

590 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) MBAINT/UPDA=(MBAINT.UPD,MBAINT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

```
;  0001  0     %title '*** MCHK730 machine check formatter'
;  0002  0     module mchk730 (                        ! Machine check formatter for NEBULA
;  0003  0                     ident = '06-07'
;  0004  0                     ) =
;  0005  0     !
;  0006  0     ! COPYRIGHT (c) 1979, 1981, 1983
;  0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;  0008  0     !
;  0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;  0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;  0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;  0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;  0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;  0014  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;  0015  0     ! REMAIN IN DEC.
;  0016  0     !
;  0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;  0018  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;  0019  0     ! CORPORATION.
;  0020  0     !
;  0021  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;  0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;  0023  0     !
;  0024  0     !++
;  0025  0     ! FACILITY:
;  0026  0     !
;  0027  0     !       DIAGNOSTIC SUPERVISOR
;  0028  0     !
;  0029  0     ! ABSTRACT:
;  0030  0     !
;  0031  0     !       Provide for the formatting of NEBULA Machine check logout.
;  0032  0     !
;  0033  0     ! AUTHOR:
;  0034  0     !
;  0035  0     !       Roger Riggs, CREATION DATE: 20-AUG-1979
;  0036  0     !
;  0037  0     ! MODIFIED BY:
;  0038  0     !       02      - Roger Riggs, 20-Jun-1980
;  0039  0     !               Modified CHR$MCHK parameters from EXCEPT and updated
;  0040  0     !               to reflect change in the machine check logout for NEBULA.
;  0041  0     !
;  0042  0     !       03      - Dave Butenhof, 19-Nov-1980, version 6.2
;  0043  0     !               Correct signal vector offsets to skip signal vector length
;  0044  0     !               and exception code (DS$_MCHK) instead of printing them
;  0045  0     !               as part of logout.
;  0046  0     !
;  0047  0     !       04      - Dave Butenhof, 28-Apr-1981, version 6.4
;  0048  0     !               Change messages to mixed case, correct spelling errors.
;  0049  0     !
;  0050  0     !       05      - Dave Butenhof, 02-Jun-1981, version 6.4
;  0051  0     !               Change library specification for flexibility
;  0052  0     !
;  0053  0     !       06      - Jack Stansbury, 24-Mar-1982, Version 6.?
;  0054  0     !               Added Library statements for Lib and $DS. Also added typecoding.
;  0055  0     !
;  0056  0     !       07      Bob Bergazzi    May 17, 1983    Version 6.11
;  0057  0     !               Changed the order of searching libraries.
```

```
;    0058  0     !--
;    0059  0
;    0060  1     begin
;    0061  1     !
;    0062  1     ! TABLE OF CONTENTS:
;    0063  1     !
;    0064  1
;    0065  1     linkage
;    0066  1         quick = jsb;
;    0067  1
;    0068  1     forward routine
;    0069  1         chr$mchk : novalue;                     ! FORMATTER FOR MACHINE LOGOUT
;    0070  1
;    0071  1     forward routine
;    0072  1         dsr$fault_clear : quick novalue;        ! RESET MACHINE CHECK
;    0073  1
;    0074  1     !
;    0075  1     ! INCLUDE FILES:
;    0076  1     !
;    0077  1
;    0078  1     Library '$Diag';                !                                                   [07]
;    0079  1
;    0080  1     Library '$DS';                  !                                                   [07]
;    0081  1
;    0082  1     Library 'Sys$Library:Lib';      !                                                   [07]
;    0083  1
;    0084  1     !
;    0085  1     ! MACROS
;    0086  1     !
;    0087  1
;    0088  1     $DS_TypeDef;                    ! Define typecodes                                  [06]
;    0089  1
;    0090  1     !
;    0091  1     ! EXTERNAL REFERENCES:
;    0092  1     !
;    0093  1
;    0094  1     external
;    0095  1         Apslmne         :       Addressing_Mode (Long_Relative),    ! PSL MNEMONICS
;    0096  1         Modelst         :       Addressing_Mode (Long_Relative),    ! PROCESSOR MODE DECODE LIST
;    0097  1         DS$GB_TypeCode  : Byte Addressing_Mode (Long_Relative);     ! The typecode number byte  [06]
;    0098  1
;    0099  1     !
;    0100  1     ! PSECT DEFINITIONS:
;    0101  1     !
;    0102  1
;    0103  1     psect
;    0104  1         plit = data ( share, addressing_mode (long_relative));
;    0105  1
;    0106  1     psect
;    0107  1         code = code ( read, execute, share);
;    0108  1
```

```
;    0109  1      %sbttl 'CHK$MCHK'
;    0110  1
;    0111  1      global routine chr$mchk (signal, hdrprint, txtprint) : novalue =
;    0112  1
;    0113  1      !++
;    0114  1      ! FUNCTIONAL DESCRIPTION:
;    0115  1      !
;    0116  1      !       Using the Machine logout information on the stack
;    0117  1      !       and the type code format each longword of the logout area.
;    0118  1      !
;    0119  1      ! FORMAL PRAMETERS:
;    0120  1      !
;    0121  1      !       SIGNAL:   Address of signal array for machine check
;    0122  1      !       HDRPRINT: Address of print routine to print hdr's
;    0123  1      !       TXTPRINT: Address of print routine to print text
;    0124  1      !
;    0125  1      !       Signal array is
;    0126  1      !       CONDITION Value DS$_MCHK
;    0127  1      !       COUNT   Bytes pushed by machine check
;    0128  1      !       TYPE    Type code of machine check
;    0129  1      !       P1      1st parameter
;    0130  1      !       P2      2nd parameter
;    0131  1      !       PC      PC of machine check
;    0132  1      !       PSL     PSL at time of machine check
;    0133  1      !
;    0134  1      ! IMPLICIT INPUTS:
;    0135  1      !
;    0136  1      !       NONE
;    0137  1      !
;    0138  1      ! IMPLICIT OUTPUTS:
;    0139  1      !
;    0140  1      !       NONE
;    0141  1      !
;    0142  1      ! COMPLETION CODES:
;    0143  1      !
;    0144  1      !       N/A (No value is returned)
;    0145  1      !
;    0146  1      ! SIDE EFFECTS:
;    0147  1      !
;    0148  1      !       The machine check error status is cleared
;    0149  1      !--
```

```
;    0150  2          begin
;    0151  2
;    0152  2          bind
;    0153  2              t_mchk = $ascic ('!/!/?? Machine check exception through vector: 04(X)!/'),     !                [04]
;    0154  2              t_count = $ascic ('count:!_!_!_!XL(X)!/'),                                                    [04]
;    0155  2              t_type = $ascic ('Machine check type:!_!XL(X)!_; !AC!/'),       !                             [04]
;    0156  2              t_va = $ascic ('Virtual address:!_!XL(X)!/'),                                                 [04]
;    0157  2              t_phy = $ascic ('Physical address:!_!XL(X)!/'),                                               [04]
;    0158  2              t_tb = $ascic ('TB entry:!_!_!XL(X)!/'),                                                      [04]
;    0159  2              t_pte = $ascic ('PTE address:!_!_!XL(X)!/'),                                                  [04]
;    0160  2              t_mcsr = $ascic ('Memory CSR:!_!_!XL(X)!/'),                                                  [04]
;    0161  2              t_int = $ascic ('Interrupt identifier:!_!XL(X)!/'),     !                                     [04]
;    0162  2              t_p1 = $ascic ('Error parameter 1:!_!XL(X)!/'),                                               [04]
;    0163  2              t_p2 = $ascic ('Error parameter 2:!_!XL(X)!/'),                                               [04]
;    0164  2              t_sub = $ascic ('Subtype:!_!_!_!XL(X)!/'),                                                    [04]
;    0165  2              t_ecc = $ascic ('PFN/Syndrome:!_!_!XL(X)!/'),                                                 [04]
;    0166  2              t_row = $ascic ('FPA row number!_!_!XL(X)!/'),                                                [04]
;    0167  2              t_unknown = $ascic ('Unknown machine check type'),      !                                     [04]
;    0168  2              list_type = uplit long(                                                                       [04]
;    0169  2                              $ascic('micro code should not be here'),       !                             [04]
;    0170  2                              $ascic('translation buffer parity error'),     !                             [04]
;    0171  2                              $ascic('undefined interrupt identifier'),      !                             [04]
;    0172  2                              $ascic('illegal memory CSR'),                                                [04]
;    0173  2                              $ascic('fast interrupt without support'),      !                             [04]
;    0174  2                              $ascic('FPA parity error'),                                                  [04]
;    0175  2                              $ascic('error on SPTE read'),                                                [04]
;    0176  2                              $ascic('uncorrectable ECC error'),             !                             [04]
;    0177  2                              $ascic('reference to non-existant memory'),    !                             [04]
;    0178  2                              $ascic('unaligned or non-longword ref to I/O space'),   !                    [04]
;    0179  2                              $ascic('illegal I/O space address'),           !                             [04]
;    0180  2                              $ascic('illegal UNIBUS reference'),            !                             [04]
;    0181  2                              t_unknown,
;    0182  2                              t_unknown,
;    0183  2                              t_unknown,
;    0184  2                              'illegal UNIBUS reference',                                                  [04]
;    0185  2                              t_unknown,
;    0186  2                              t_unknown,
;    0187  2                              t_unknown,
;    0188  2                              t_unknown) : vector [, long],
;    0189  2                  list_p1 = uplit long(t_sub, t_va, t_int, t_va,
;    0190  2                              t_p1,t_row,t_phy,t_ecc,
;    0191  2                              t_phy,t_phy,t_phy,t_phy,
;    0192  2                              t_p1, t_row, t_phy, t_ecc,
;    0193  2                              t_phy, t_phy, t_phy, t_phy,
;    0194  2                              t_p1, t_p1, t_p1, t_p1) : vector [, long],      !
;    0195  2                  list_p2 = uplit long(t_p2, t_tb, t_p2, t_mcsr,
;    0196  2                              t_p2,t_p2,t_mcsr,t_p2,
;    0197  2                              t_p2,t_p2,t_p2,t_p2,
;    0198  2                              t_p2, t_p2, t_mcsr, t_p2,
;    0199  2                              t_p2, t_p2, t_p2, t_p2, t_p2,
;    0200  2                              t_p2, t_p2, t_p2) : vector [, long];    !
;    0201  2
;    0202  2          local
;    0203  2              index,
;    0204  2              buffer : vector [128, byte];
;    0205  2
;    0206  2          map
```

```
;    0207  2              signal : ref vector [, long];
;    0208  2
;    0209  2          DS$GB_TypeCode = DS$K_Type_Exception_Head;          ! Print the exception header message                [06]
;    0210  2          (.hdrprint) (t_mchk);
;    0211  2          DS$GB_TypeCode = DS$K_Type_Exception;               ! Print the exception text messages                [06]
;    0212  2          (.txtprint) (t_count, .signal [2]);
;    0213  2          index = (if .signal [3] gtr 13 then 2 else .signal [3]);
;    0214  2          (.txtprint) (t_type, .signal [3], .list_type [.index]);
;    0215  2          (.txtprint) (.list_p1 [.index], .signal [4]);
;    0216  2          (.txtprint) (.list_p2 [.index], .signal [5]);
;    0217  2          DS$GB_TypeCode = 0;                                 ! Clear out the typecode byte                      [06]
;    0218  1          end;


                                                                .TITLE  MCHK730 *** MCHK730 machine check formatter
                                                                .IDENT  \06-07\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

65 6E 69 68 63 61 4D 20 3F 3F 2F 21 2F 21 36  00000 P.AAA:  .ASCII  \6!/!/?? Machine check exception through \  ;
6F 69 74 70 65 63 78 65 20 6B 63 65 68 63 20  0000F                                                            ;
                  20 68 67 75 6F 72 68 74 20 6E  0001E                                                            ;
2F 21 29 58 28 34 30 20 3A 72 6F 74 63 65 76  00028          .ASCII  \vector: 04(X)!/\                          ;
58 21 5F 21 5F 21 5F 21 3A 74 6E 75 6F 63 14  00037 P.AAB:   .ASCII  <20>\count:!_!_!_!XL(X)!/\                   ;
                           2F 21 29 58 28 4C  00046                                                            ;
20 6B 63 65 68 63 20 65 6E 69 68 63 61 4D 24  0004C P.AAC:   .ASCII  \$Machine check type:!_!XL(X)!_; !AC!/\       ;
5F 21 29 58 28 4C 58 21 5F 21 3A 65 70 79 74  00058                                                            ;
                        2F 21 43 41 21 20 3B  0006A                                                            ;
73 65 72 64 64 61 20 6C 61 75 74 72 69 56 1A  00071 P.AAD:   .ASCII  <26>\Virtual address:!_!XL(X)!/\           ;
                  2F 21 29 58 28 4C 58 21 5F 21 3A 73  00080                                                   ;
65 72 64 64 61 20 6C 61 63 69 73 79 68 50 1B  0008C P.AAE:   .ASCII  <27>\Physical address:!_!XL(X)!/\          ;
                  2F 21 29 58 28 4C 58 21 5F 21 3A 73  0009B                                                   ;
21 5F 21 5F 21 3A 79 72 74 6E 65 20 42 54 15  000A8 P.AAF:   .ASCII  <21>\TB entry:!_!_!XL(X)!/\               ;
                        2F 21 29 58 28 4C 58  000B7                                                            ;
5F 21 3A 73 73 65 72 64 64 61 20 45 54 50 18  000BE P.AAG:   .ASCII  <24>\PTE address:!_!_!XL(X)!/\            ;
                  2F 21 29 58 28 4C 58 21 5F 21  000CD                                                         ;
21 5F 21 3A 52 53 43 20 79 72 6F 6D 65 4D 17  000D7 P.AAH:   .ASCII  <23>\Memory CSR:!_!_!XL(X)!/\             ;
                  2F 21 29 58 28 4C 58 21 5F  000E6                                                            ;
6E 65 64 69 20 74 70 75 72 72 65 74 6E 49 1F  000EF P.AAI:   .ASCII  <31>\Interrupt identifier:!_!XL(X)!/\      ;
29 58 28 4C 58 21 5F 21 3A 72 65 69 66 69 74  000FE                                                            ;
                                    2F 21  0010D                                                               ;
65 74 65 6D 61 72 61 70 20 72 6F 72 72 45 1C  0010F P.AAJ:   .ASCII  <28>\Error parameter 1:!_!XL(X)!/\        ;
   2F 21 29 58 28 4C 58 21 5F 21 3A 31 20 72  0011E                                                           ;
65 74 65 6D 61 72 61 70 20 72 6F 72 72 45 1C  0012C P.AAK:   .ASCII  <28>\Error parameter 2:!_!XL(X)!/\        ;
   2F 21 29 58 28 4C 58 21 5F 21 3A 32 20 72  0013B                                                           ;
5F 21 5F 21 5F 21 3A 65 70 79 74 62 75 53 16  00149 P.AAL:   .ASCII  <22>\Subtype:!_!_!_!XL(X)!/\             ;
                        2F 21 29 58 28 4C 58 21  00158                                                         ;
21 3A 65 6D 6F 72 64 6E 79 53 2F 4E 46 50 19  00160 P.AAM:   .ASCII  <25>\PFN/Syndrome:!_!_!XL(X)!/\           ;
                  2F 21 29 58 28 4C 58 21 5F 21 5F  0016F                                                      ;
72 65 62 6D 75 6E 20 77 6F 72 20 41 50 46 1A  0017A P.AAN:   .ASCII  <26>\FPA row number!_'_!XL(X)!/\          ;
                  2F 21 29 58 28 4C 58 21 5F 21 5F 21  00189                                                   ;
6E 69 68 63 61 6D 20 6E 77 6F 6E 6B 6E 55 1A  00195 P.AAO:   .ASCII  <26>\Unknown machine check type\           ;
                  65 70 79 74 20 6B 63 65 68 63 20 65  001A4                                                   ;
6F 68 73 20 65 64 6F 63 20 6F 72 63 69 6D 1D  001B0 P.AAQ:   .ASCII  <29>\micro code should not be here\        ;
65 72 65 68 20 65 62 20 74 6F 6E 20 64 6C 75  001BF                                                           ;
75 62 20 6E 6F 69 74 61 6C 73 6E 61 72 74 1F  001CE P.AAR:   .ASCII  <31>\translation buffer parity error\     ;
72 72 65 20 79 74 69 72 61 70 20 72 65 66 66  001DD                                                           ;
```

```
                                                                72 6F 001EC
65 74 6E 69 20 64 65 6E 69 66 65 64 6E 75 1E 001EE P.AAS:  .ASCII  <30>\undefined interrupt identifier\
65 69 66 69 74 6E 65 64 69 20 74 70 75 72 72 001FD
                                                                72 0020C
79 72 6F 6D 65 6D 20 6C 61 67 65 6C 6C 69 12 0020D P.AAT:  .ASCII  <18>\illegal memory CSR\
                                             52 53 43 20 0021C
74 70 75 72 72 65 74 6E 69 20 74 73 61 66 1E 00220 P.AAU:  .ASCII  <30>\fast interrupt without support\
72 6F 70 70 75 73 20 74 75 6F 68 74 69 77 20 0022F
                                                   74 0023E
72 72 65 20 79 74 69 72 61 70 20 41 50 46 10 0023F P.AAV:  .ASCII  <16>\FPA parity error\
                                                72 6F 0024E
20 45 54 50 53 20 6E 6F 20 72 6F 72 72 65 12 00250 P.AAW:  .ASCII  <18>\error on SPTE read\
                                          64 61 65 72 0025F
20 65 6C 62 61 74 63 65 72 72 6F 63 6E 75 17 00263 P.AAX:  .ASCII  <23>\uncorrectable ECC error\
                     72 6F 72 72 65 20 43 43 45 00272
6E 20 6F 74 20 65 63 6E 65 72 65 66 65 72 20 0027B P.AAY:  .ASCII  \ reference to non-existant memory\
6D 65 6D 20 74 6E 61 74 73 69 78 65 2D 6E 6F 0028A
                                    79 72 6F 00299
6E 20 72 6F 20 64 65 6E 67 69 6C 61 6E 75 2A 0029C P.AAZ:  .ASCII  \*unaligned or non-longword ref to I/O sp\
66 65 72 20 64 72 6F 77 67 6E 6F 6C 2D 6E 6F 002AB
                  70 73 20 4F 2F 49 20 6F 74 20 002BA
                                       65 63 61 002C4            .ASCII  \ace\
70 73 20 4F 2F 49 20 6C 61 67 65 6C 6C 69 19 002C7 P.ABA:  .ASCII  <25>\illegal I/O space address\
               73 73 65 72 64 64 61 20 65 63 61 002D6
53 55 42 49 4E 55 20 6C 61 67 65 6C 6C 69 18 002E1 P.ABB:  .ASCII  <24>\illegal UNIBUS reference\
                  65 63 6E 65 72 65 66 65 72 20 002F0
                                                002FA            .BLKB   2
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 002FC P.AAP:  .ADDRESS P.AAQ, P.AAR, P.AAS, P.AAT, P.AAU, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00314          P.AAV, P.AAW, P.AAX, P.AAY, P.AAZ, P.ABA, -
                    00000000' 00000000' 00000000' 0032C                    P.ABB, T_UNKNOWN, T_UNKNOWN, T_UNKNOWN
20 53 55 42 49 4E 55 20 6C 61 67 65 6C 6C 69 00338          .ASCII  \illegal UNIBUS reference\
                     65 63 6E 65 72 65 66 65 72 00347
                  00000000' 00000000' 00000000' 00000000' 00350          .ADDRESS T_UNKNOWN, T_UNKNOWN, T_UNKNOWN, -
                                                                                 T_UNKNOWN
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00360 P.ABC:  .ADDRESS T_SUB, T_VA, T_INT, T_VA, T_P1, T_ROW, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00378                 T_PHY, T_ECC, T_PHY, T_PHY, T_PHY, T_PHY, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00390                 T_P1, T_ROW, T_PHY, T_ECC, T_PHY, T_PHY, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 003A8                 T_PHY, T_PHY, T_P1, T_P1, T_P1, T_PT
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 003C0 P.ABD:  .ADDRESS T_P2, T_TB, T_P2, T_MCSR, T_P2, T_P2, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 003D8                 T_MCSR, T_P2, T_P2, T_P2, T_P2, T_P2, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 003F0                 T_P2, T_P2, T_MCSR, T_P2, T_P2, T_P2, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00408                 T_P2, T_P2, T_P2, T_P2, T_P2, T_P2
```

```
T_MCHK=     P.AAA
T_COUNT=    P.AAB
T_TYPE=     P.AAC
T_VA=       P.AAD
T_PHY=      P.AAE
T_TB=       P.AAF
T_PTE=      P.AAG
T_MCSR=     P.AAH
T_INT=      P.AAI
T_P1=       P.AAJ
T_P2=       P.AAK
T_SUB=      P.AAL
T_ECC=      P.AAM
T_ROW=      P.AAN
```

```
                                                T_UNKNOWN=          P.AAO
                                                LIST_TYPE=          P.AAP
                                                LIST_P1=            P.ABC
                                                LIST_P2=            P.ABD
                                                        .EXTRN  APSLMNE, MODELST
                                                        .EXTRN  DS$GB_TYPECODE

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                        003C 00000      .ENTRY  CHR$MCHK, Save R2,R3,R4,R5                      ; 0111
                55 00000000G    EF   9E 00002           MOVAB   DS$GB_TYPECODE, R5                              ;
                54 00000000'    EF   9E 00009           MOVAB   T_MCHK, R4                                      ;
                5E        80    AE   9E 00010           MOVAB   -128(SP), SP                                    ;
                65              0B   90 00014           MOVB    #11, DS$GB_TYPECODE                             ; 0209
                                54   DD 00017           PUSHL   R4                                              ; 0210
            08  BC              01   FB 00019           CALLS   #1, @HDRPRINT                                   ;
                65              0C   90 0001D           MOVB    #12, DS$GB_TYPECODE                             ; 0211
                53        04    AC   D0 00020           MOVL    SIGNAL, R3                                      ; 0212
                          08    A3   DD 00024           PUSHL   8(R3)                                           ;
                          37    A4   9F 00027           PUSHAB  T_COUNT                                         ;
            0C  BC              02   FB 0002A           CALLS   #2, @TXTPRINT                                   ;
                0D        0C    A3   D1 0002E           CMPL    12(R3), #13                                     ; 0213
                                05   15 00032           BLEQ    1$                                              ;
                52              02   D0 00034           MOVL    #2, INDEX                                       ;
                                04   11 00037           BRB     2$                                              ;
                52        0C    A3   D0 00039  1$:      MOVL    12(R3), INDEX                                   ;
                        02FC  C442   DD 0003D  2$:      PUSHL   LIST_TYPE[INDEX]                                ; 0214
                          0C    A3   DD 00042           PUSHL   12(R3)                                          ;
                          4C    A4   9F 00045           PUSHAB  T_TYPE                                          ;
            0C  BC              03   FB 00048           CALLS   #3, @TXTPRINT                                   ;
                          10    A3   DD 0004C           PUSHL   16(R3)                                          ; 0215
                        0360  C442   DD 0004F           PUSHL   LIST_P1[INDEX]                                  ;
            0C  BC              02   FB 00054           CALLS   #2, @TXTPRINT                                   ;
                          14    A3   DD 00058           PUSHL   20(R3)                                          ; 0216
                        03C0  C442   DD 0005B           PUSHL   LIST_P2[INDEX]                                  ;
            0C  BC              02   FB 00060           CALLS   #2, @TXTPRINT                                   ;
                                05   94 00064           CLRB    DS$GB_TYPECODE                                  ; 0217
                                     04 00066           RET                                                     ; 0218

; Routine Size:  103 bytes,    Routine Base:  CODE + 0000

;   0219  1
```

```
;    0220  1      %sbttl 'DSR$FAULT_CLEAR'
;    0221  1
;    0222  1      global routine dsr$fault_clear : quick novalue =
;    0223  1
;    0224  1      !++
;    0225  1      ! FUNCTIONAL DESCRIPTION:
;    0226  1      !
;    0227  1      !       This routine is called to clear the processor dependent
;    0228  1      !       error status.
;    0229  1      !
;    0230  1      ! FORMAL PARAMETERS:
;    0231  1      !
;    0232  1      !       NONE
;    0233  1      !
;    0234  1      ! IMPLICIT INPUTS:
;    0235  1      !
;    0236  1      !       NONE
;    0237  1      !
;    0238  1      ! IMPLICIT OUTPUTS:
;    0239  1      !
;    0240  1      !       NONE
;    0241  1      !
;    0242  1      ! COMPLETION CODES:
;    0243  1      !
;    0244  1      !       NONE
;    0245  1      !
;    0246  1      ! SIDE EFFECTS:
;    0247  1      !
;    0248  1      !       Machine check status is cleared
;    0249  1      !--
```

```
;    0250  1
;    0251  2          begin
;    0252  2
;    0253  2          literal
;    0254  2              pr$_mcesr = %x'26';
;    0255  2
;    0256  2          builtin
;    0257  2              mtpr;
;    0258  2
;    0259  2          mtpr (%ref (0), pr$_mcesr);
;    0260  1          end;


                                26            00  DA 00000 DSR$FAULT_CLEAR::
                                                              MTPR    #0, #38                 ; 0259
                                                  05 00003    RSB                             ; 0260

; Routine Size:  4 bytes,    Routine Base:  CODE + 0067

;    0261  1
;    0262  1      end
;    0263  1
;    0264  0      eludom
```

```
                                PSECT SUMMARY

      Name                      Bytes                           Attributes

DATA                             1056  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
CODE                              107  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


                            Library Statistics

                                  --------  Symbols  --------      Pages       Processing
      File                        Total     Loaded   Percent      Mapped      Time

DRB1:[DS.WORK]DIAG.L32;265          784          2         0          85       00:00.1
DRB1:[DS.WORK]DS.L32;159            653          0         0          42       00:00.2
SYS$SYSROOT:[SYSLIB]LIB.L32;7     18017          4         0         975       00:04.8
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE MCHK730

; Size:          107 code + 1056 data bytes
; Run Time:        00:12.6
; Elapsed Time:    00:27.1
; Lines/CPU Min:    1254
; Lexemes/CPU-Min: 18579
; Memory Used:  97 pages
; Compilation Complete
```

Table of contents

(2) 135 ALLOCATE MEMORY AND CONDITIONALLY WAIT
(2) 252 ALLOCATE NONPAGED DYNAMIC MEMORY
(2) 310 GENERAL ALLOCATE MEMORY SUBROUTINE
(2) 348 DEALLOCATE NONPAGED DYNAMIC MEMORY
(2) 398 CHECK BLOCK PARAMETERS SUBROUTINE
(2) 414 GENERAL DEALLOCATION SUBROUTINE
(2) 456 MEMPOOL$INIT ;USER MODE MEMORY I ALIZATION

```
0000     1          .TITLE  MEMALC  *** MEMALC dynamic memory allocation
0000     2          .IDENT  /07-06/
0000     3          .NLIST  CND
0000     4
0000     5 ;
0000     6 ;*****************************************************************************
0000     7 ;*                                                                           *
0000     8 ;*  Copyright (c) 1977, 1982                                                 *
0000     9 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    12 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    14 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
C000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
C000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ; D. N. CUTLER 3-AUG-76
0000    28 ;       V0206   LMK0003         LEN KAWELL                  02-NOV-1979
0000    29 ;               ADD RESOURCE REPORTING TO SHARED MEMORY DEALLOCATION.
0000    30 ;
0000    31 ;
0000    32 ;
0000    33 ;               N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    34 ;       01      DIAGNOSTIC SUPERVISOR INTEGRATION
0000    35 ;               Roger Riggs     05-Jul-78                   (ESSAA-4.03)
0000    36 ;       02      Removed REMQUE/INSQUE instructions and made
0000    37 ;               corresponding DSBINT/ENBINT macros execute only if in S/A
0000    38 ;               in EXE$ALONONPAGED/EXE$DEANONPAGED
0000    39 ;               Roger Riggs                 20-NOV-1979
0000    40 ;       03      Removed L^ from REF to DS$GL_MEMSIZE
0000    41 ;               Dave Butenhof   15-may-1980, Version 5.4
0000    42 ;       04      Create .UPD file to modify VMS V2.0 source to supervisor
0000    43 ;               environment.
0000    44 ;
0000    .1 ;       05      - Jack Stansbury, 10-Nov-1981, Version 6.-
-1      0000    46 ;               Fixed truncation errors by replacing all W^ with L^.
0000    47 ;               Added .LIBRARY statements for $DIAG and $DS.
0000    48 ;
0000    49 ;       06      - Dave Butenhof, 05-Feb-1982, version 6.6
0000    50 ;               Make EXE$GL_SPLITADR global for access by SHOW MEMORY,
0000    51 ;               and add DS$GL_LOOKASIDE to remember start of lookaside
0000    52 ;               list (if any).
0000    53 ;
0000    54 ;       V0205   RIH0031         RICHARD I. HUSTVEDT     08-AUG-1979
0000    55 ;               ADD ALLOCATE ROUTINE FOR JIB AND CHANGE USE OF LOOKASIDE
0000    56 ;               LIST TO SATISFY IRP-SIZE AND SMALLER REQUESTS.
0000    57 ;
```

```
                                0000     58 ;       V0204   KDM0034          KATHLEEN D. MORSE       22-MAY-1979
                                0000     59 ;               ADD ASSUMPTIONS FOR COMMON EVENT BLOCK SIZES.
                                0000     60 ;
                                0000     61 ;       V0203   LMK0002          LEN KAWELL              18-MAY-1979
                                0000     62 ;               ADDED DEALLOCATION OF ''SPECIAL'' TYPES OF NON-PAGED
                                0000     63 ;               POOL BLOCKS.
                                0000     64 ;
                                0000     65 ;       V0202   LMK0001          LEN KAWELL              25-FEB-1979
                                0000     66 ;               ADDED SHARED MEMORY POOL ALLOCATION/DEALLOCATION
                                0000     67 ;
                                0000     68 ; DYNAMIC MEMORY ALLOCATION
```

```
                        0000     70 ;
                        0000     71 ; LIBRARYS
                        0000     72 ;
                        0000     73         .Library        /Sys$Library:Lib/
                        0000     74         .LIBRARY        /$DS/           ;                                       [05]
                        0000     75         .LIBRARY        /$DIAG/         ;                                       [05]
                        0000     76
                        0000     77 ;
                        0000     78 ; MACRO LIBRARY CALLS
                        0000     79 ;
                        0000     80
                        0000     81         $DS_DSADEF                              ;DEFINE BITS IN DSA$GL_FLAGS
                        0000     82         $CEBDEF                         ;DEFINE COMMON EVENT BLOCKS
                        0000     83         $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                        0000     84         $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000     85         $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000     86         $JIBDEF                         ;DEFINE JIB OFFSETS
                        0000     87         $PCBDEF                         ;DEFINE PCB OFFSETS
                        0000     88         $PQBDEF                         ;DEFINE PQB OFFSETS
                        0000     89         $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     90         $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000     91         $SHBDEF                         ;DEFINE SHARED MEM CONTROL BLOCK
                        0000     92         $SHDDEF                         ;DEFINE SHARED MEM DATAPAGE
                        0000     93         $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                        0000     94         $TQEDEF                         ;DEFINE TQE OFFSETS
                        0000     95
                        0000     96 ;
                        0000     97 ; LOCAL SYMBOLS
                        0000     98 ;
                        0000     99 ; ALLOCATION GRANULARITY MASK
                        0000    100 ;
                        0000    101
                0000000F 0000   102 MASK=^XF                                    ;16 BYTE ALLOCATION GRANULARITY
                        0000    103
                00000000 0000   104         .PSECT  WORK,   NOSHR,  NOEXE,  WRT,    LONG
                        0000    105 ;
                        0000    106 ;OWN STORAGE
                        0000    107 ;
                        0000    108
                        0000    109         .ALIGN  LONG
                        0000    110 EXE$GL_NONPAGED:
                0000000B 0000   111         .LONG   11                          ;DISABLE FORK IPL
                00000000 0004   112         .LONG   0                           ;ADDRESS OF FIRST FREE BLOCK
                00000000 0008   113         .LONG   0                           ;NO BYTES IN BLOCK
                        000C    114
                        000C    115 EXE$GL_SPLITADR::                           ;LOOKASIDE I/O PACKET LIST SPLIT ADDRESS
                00000000 000C   116         .LONG   0                           ;ADDRESS OF LOWEST IRP
                        0010    117 DS$GL_LOOKASIDE::                           ; Address of lookaside start
                00000000 0010   118         .LONG   0
                        0014    119
                        0014    120
                        0014    121 ;
                        0014    122 ;I/O PACKET LOOK ASIDE LISTHEAD
                        0014    123 ;
                        0014    124
                        0014    125 IOC$GL_IRPFL::
00000014'00000014'      0014    126         .LONG   IOC$GL_IRPFL,IOC$GL_IRPFL       ;BACKWARD LINK
```

```
                    00000018  001C   127 IOC$GL_IRPBL=IOC$GL_IRPFL+4
                              001C   128
                              001C   129
                          00000000   130          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                              0000   131          MODNAM  MEMALC
                              0007   132 T_CGF:
                 4F 4E 4E 41 43 00' 0007   133          .ASCIC  .CANNOT GET FREE CORE.
                 20 54 45 47 20 54  000D
                 43 20 45 45 52 46  0013
                       45 52 4F     0019
                             14     0007
```

```
                     001C   135          .SBTTL  ALLOCATE MEMORY AND CONDITIONALLY WAIT
                  00000000  136 .PSECT  CODE     SHR,    EXE,    NOWRT,  BYTE
                     0000   137
                     0000   138 ;+
                     0000   139 ; EXE$ALLOCBUF - ALLOCATE BUFFERED I/O BUFFER AND CONDITIONALLY WAIT
                     0000   140 ;
                     0000   141 ; THIS ROUTINE IS CALLED TO ALLOCATE A BUFFERED I/O BUFFER. IF SUFFICIENT
                     0000   142 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
                     0000   143 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   144 ;
                     0000   145 ; EXE$ALLOCCEB - ALLOCATE COMMON EVENT BLOCK AND CONDITIONALLY WAIT
                     0000   146 ;
                     0000   147 ; THIS ROUTINE IS CALLED TO ALLOCATE A COMMON EVENT BLOCK. IF SUFFICIENT
                     0000   148 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
                     0000   149 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   150 ;
                     0000   151 ; EXE$ALLOCJIB - ALLOCATE JOB INFORMATION BLOCK AND CONDITIONALLY WAIT
                     0000   152 ;
                     0000   153 ; THIS ROUTINE IS CALLED TO ALLOCATE A JOB INFORMATION BLOCK. IF SUFFICIENT
                     0000   154 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
                     0000   155 ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   156 ;
                     0000   157 ; EXE$ALLOCIRP - ALLOCATE I/O REQUEST PACKET AND CONDITIONALLY WAIT
                     0000   158 ;
                     0000   159 ; THIS ROUTINE IS CALLED TO ALLOCATE AN I/O PACKET. IF SUFFICIENT MEMORY
                     0000   160 ; IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
                     0000   161 ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   162 ;
                     0000   163 ; EXE$ALLOCPCB - ALLOCATE PROCESS CONTROL BLOCK AND CONDITIONALLY WAIT
                     0000   164 ;
                     0000   165 ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS CONTROL BLOCK WHEN
                     0000   166 ; CREATING A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN
                     0000   167 ; A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED DEPENDING ON THE CURRENT
                     0000   168 ; PROCESS' RESOURCE WAIT MODE.
                     0000   169 ;
                     0000   170 ; EXE$ALLOCPQB - ALLOCATE PROCESS QUOTA BLOCK AND CONDITIONALLY WAIT
                     0000   171 ;
                     0000   172 ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS QUOTA BLOCK WHEN CREATING
                     0000   173 ; A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN A RESOURCE
                     0000   174 ; WAIT STATE IS ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT
                     0000   175 ; MODE.
                     0000   176 ;
                     0000   177 ; EXE$ALLOCTQE - ALLOCATE TIME QUEUE ENTRY AND CONDITIONALLY WAIT
                     0000   178 ;
                     0000   179 ; THIS ROUTINE IS CALLED TO ALLOCATE A TIME QUEUE ENTRY. IF SUFFICIENT
                     0000   180 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
                     0000   181 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   182 ;
                     0000   183 ; INPUTS:
                     0000   184 ;
                     0000   185 ;       R4 = NORMALLY CURRENT PROCESS PCB ADDRESS, BUT NOT REQUIRED.
                     0000   186 ;
                     0000   187 ; IF ENTRY AT EXE$ALLOCBUF, THEN
                     0000   188 ;
                     0000   189 ;       R1 = SIZE OF REQUESTED BUFFER IN BYTES.
                     0000   190 ;
                     0000   191 ; OUTPUTS:
```

```
                                0000   192 ;
                                0000   193 ;       R0 = LOW BIT CLEAR IF ALLOCATION FAILURE WITH CALLING IPL PRESERVED.
                                0000   194 ;
                                0000   195 ;               R0 = SS$_INSFMEM = INSUFFICIENT MEMORY AVAILABLE TO ALLOCATE
                                0000   196 ;                       BUFFER.
                                0000   197 ;
                                0000   198 ;       R0 = LOW BIT SET IF SUCCESSFUL ALLOCATION WITH:
                                0000   199 ;
                                0000   200 ;               R1 = SIZE OF REQUESTED BUFFER IN BYTES.
                                0000   201 ;               R2 = ADDRESS OF ALLOCATED BUFFER WITH SIZE AND TYPE FIELDS
                                0000   202 ;                       FILLED IN.
                                0000   203 ;
                                0000   204 ;               AND IPL SET TO AST DELIVERY LEVEL.
                                0000   205 ;
                                C000   206 ;       R4 = ORIGINAL R4 OR CURRENT PCB IF A WAIT OCCURRED.
                                0000   207 ;-
                                0000   208
                                0000   209                .ENABL  LSB
                                0000   210 EXE$ALLOCBUF::                                ;ALLOCATE BUFFERED I/O BUFFER
                          13    DD 0000   211                PUSHL   #DYN$C_BUFIO            ;SET DATA STRUCTURE TYPE
                          28    11 0002   212                BRB     20$                     ;
                                0004   213 EXE$ALLOCCEB::                                ;ALLOCATE COMMON EVENT BLOCK
                          04    DD 0004   214                PUSHL   #DYN$C_CEB              ;SET DATA STRUCTURE TYPE
                                0006   215                ASSUME  CEB$C_LENGTH LE IRP$C_LENGTH ;IRP SIZE PACKETS ARE ALLOCATED
                                0006   216                ASSUME  CEB$C_SLAVLNG LE IRP$C_LENGTH ;FOR CEB'S TO LIMIT FRAGMENTATION
                          20    11 0006   217                BRB     10$                     ;
                                0008   218 EXE$ALLOCJIB::                                ;ALLOCATE JOB INFORMATION BLOCK - COND WAIT
                          2F    DD 0008   219                PUSHL   #DYN$C_JIB              ;SET STRUCTURE TYPE
                51  006C     8F    3C 000A   220                MOVZWL  #JIB$C_LENGTH,R1        ;AND LENGTH OF BLOCK
                          1B    11 000F   221                BRB     20$                     ;MERGE WITH COMMON ALLOCATE CODE
                                0011   222 EXE$ALLOCIRP::                                ;ALLOCATE I/O PACKET - CONDITIONAL WAIT
                          0A    DD 0011   223                PUSHL   #DYN$C_IRP              ;SET DATA STRUCTURE TYPE
                          13    11 0013   224                BRB     10$                     ;
                                0015   225 EXE$ALLOCPCB::                                ;ALLOCATE PROCESS CONTROL BLOCK
                          0C    DD 0015   226                PUSHL   #DYN$C_PCB              ;SET DATA STRUCTURE TYPE
              51    8C    8F    9A 0017   227                MOVZBL  #PCB$C_LENGTH,R1        ;AND STRUCTURE SIZE
                          0F    11 001B   228                BRB     20$                     ;
                                001D   229 EXE$ALLOCPQB::                                ;ALLOCATE PROCESS QUOTA BLOCK
                          0D    DD 001D   230                PUSHL   #DYN$C_PQB              ;SET DATA STRUCTURE TYPE
              51  08C8    8F    3C 001F   231                MOVZWL  #PQB$C_LENGTH,R1        ;AND STRUCTURE SIZE
                          06    11 0024   232                BRB     20$                     ;
                                0026   233 EXE$ALLOCTQE::                                ;ALLOCATE TIME QUEUE ENTRY
                          0F    DD 0026   234                PUSHL   #DYN$C_TQE              ;SET DATA STRUCTURE TYPE
              51    60    8F    9A 0028   235 10$:           MOVZBL  #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SET SIZE OF BUFFER REQUIRED
                          7E    DC 002C   236 20$:           MOVPSL  -(SP)                   ;READ CURRENT PSL
                          51    DD 002E   237                PUSHL   R1                      ;SAVE REQUEST SIZE
                          2E    10 0030   238                BSBB    EXE$ALONONPAGED         ;ATTEMPT TO ALLOCATE PACKET
                          0A    BA 0032   239                POPR    #^M<R1,R3>              ;RETRIEVE REQUEST SIZE AND PREVIOUS IPL
                       0C 50    E9 0034   240                BLBC    R0,40$                  ;IF LBC NO PACKET ALLOCATED
                    08 A2  51    B0 0037   241                MOVW    R1,IRP$W_SIZE(R2)       ;INSERT SIZE OF ALLOCATED BLOCK
                0A  A2  04  AE    9B 003B   242                MOVZBW  4(SP),IRP$B_TYPE(R2)    ;INSERT DATA STRUCTURE TYPE
                                0040   243                                                ;AND CLEAR MISCELLANEOUS BYTE
                          8E    D5 0040   244                TSTL    (SP)+                   ;Remove Type from stack
                          05    0042   245                RSB                             ;
              50  0124    8F    3C 0043   246 40$:           MOVZWL  #SS$_INSFMEM,R0         ;SET INSUFFICIENT MEMORY
                                0048   247                ERRSUP_S        MSGADR=T_CGF    ;CANNOT GET FREE CORE
                          00    005D   248                HALT                            ;HALT FOR NOW
```

```
                    E3   11  005E   249          BRB     40$                    ;Prevent CONTINUE
                             0060   250          .DSABL  LSB
```

```
                                0060   252          .SBTTL  ALLOCATE NONPAGED DYNAMIC MEMORY
                                0060   253 ;+
                                0060   254 ; EXE$ALONONPAGED - ALLOCATE NONPAGED DYNAMIC MEMORY
                                0060   255 ;
                                0060   256 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE NONPAGED POOL.
                                0060   257 ; IF THE BLOCK IS THE SAME SIZE AS AN I/O PACKET, AN ATTEMPT IS MADE TO ALLO-
                                0060   258 ; CATE IT FROM THE LOOKASIDE LIST.
                                0060   259 ;
                                0060   260 ; INPUTS:
                                0060   261 ;
                                0060   262 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                0060   263 ;
                                0060   264 ; OUTPUTS:
                                0060   265 ;
                                0060   266 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                0060   267 ;
                                0060   268 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                0060   269 ;
                                0060   270 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                0060   271 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                0060   272 ;-
                                0060   273
                                0060   274          .ENABL  LSB
                                0060   275 EXE$ALONONPAGED::                          ;ALLOCATE NONPAGED MEMORY
                51    0F   C0   0060   276          ADDL    #MASK,R1                  ;ROUND SIZE UP TO NEXT BOUNDRY
                51    0F   CA   0063   277          BICL    #MASK,R1                  ;TRUNCATE SIZE BACK TO MULTIPLE
                      4A   13   0066   278          BEQL    20$                       ;IF EQL BAD ALLOCATION REQUEST
                      1C   E0   0068   279          BBS     #DSA$V_USER, -
          0000FE00'EF              006A
                      06         006F   280                  DSA$GL_FLAGS,15$          ;SKIP DSBINT IF IN USER MODE
                                 0070   281          DSBINT  #^X1F
                                 0076   282 15$:
           51   0060 8F   B1     0076   283          CMPW    #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SIZE EQUAL TO I/O PACKET?
                      1E   1F     007B   284          BLSSU   10$                       ;IF NEQ NO
                                 007D   285 ;+
                                 007D   286 ;       Replaced        REMQUE  @L^IOC$GL_IRPFL,R2
                                 007D   287 ;-
          00000014'EF    DE     007D   288          MOVAL   @L^IOC$GL_IRPFL,R2        ;FIRST ENTRY IN QUEUE
                      52         0083
            04   A2    52   D1   0084   289          CMPL    R2,4(R2)                  ;QUEUE EMPTY?
                      11   13     0088   290          BEQL    10$                       ;YES,
            04   B2    62   D0   008A   291          MOVL    (R2),@4(R2)               ;COPY PREDECESSOR LINK
                 50    62   D0   008E   292          MOVL    (R2),R0                   ;COPY LINK TO SUCCESSOR
          04   A0    04   A2   D0   0091   293          MOVL    4(R2),4(R0)               ;COPY PREDECESOR LINK
                 50    01   D0   0096   294          MOVL    #SS$_NORMAL,R0            ;SET SUCCESSFUL COMPLETION
                       0B   11   0099   295          BRB     40$                       ;ENABLE AND EXIT
          00000000'EF    9E     009B   296 10$:     MOVAB   L^EXE$GL_NONPAGED,R3      ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                      53         00A1
                      83   D5   00A2   297          TSTL    (R3)+                     ;Skip IPL for list
                      21   10   00A4   298          BSBB    EXE$ALLOCATE              ;ALLOCATE BLOCK
                                 00A6   299 40$:
                      1C   E0   00A6   300          BBS     #DSA$V_USER, -
          0000FE00'EF              00A8
                      03         00AD   301                  DSA$GL_FLAGS,50$          ;SKIP ENBINT IF IN USER MODE
                                 00AE   302          ENBINT
                                 00B1   303 50$:
                           05    00B1   304          RSB                               ;
```

```
                00B2   305 20$:    BUG_CHECK BADALORQSZ            ;BAD ALLOCATION REQUEST SIZE
          50    D4  00C4   306        CLRL    R0                   ;INDICATE NO BLOCK ALLOCATED
                05  00C6   307        RSB                          ;
                    00C7   308        .DSABL  LSB
```

```
                                00C7   310          .SBTTL  GENERAL ALLOCATE MEMORY SUBROUTINE
                                00C7   311 ;+
                                00C7   312 ; EXE$ALLOCATE - ALLOCATE MEMORY SUBROUTINE
                                00C7   313 ;
                                00C7   314 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM A POOL WHOSE ENTRIES
                                00C7   315 ; ARE MAINTAINED IN A MEMORY ORDER SORTED LIST.
                                00C7   316 ;
                                00C7   317 ; INPUTS:
                                00C7   318 ;
                                00C7   319 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                00C7   320 ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                                00C7   321 ;
                                00C7   322 ; OUTPUTS:
                                00C7   323 ;
                                00C7   324 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                00C7   325 ;
                                00C7   326 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                00C7   327 ;
                                00C7   328 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                00C7   329 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                00C7   330 ;-
                                00C7   331
                                00C7   332 EXE$ALLOCATE::                               ;ALLOCATE MEMORY
                     50   53   D0   00C7   333          MOVL    R3,R0                   ;COPY ADDRESS OF FIRST FREE BLOCK ADDRESS
                     52   50   D0   00CA   334 10$:     MOVL    R0,R2                   ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
                     50   62   D0   00CD   335          MOVL    (R2),R0                 ;GET ADDRESS OF NEXT FREE BLOCK
                          1C   13   00D0   336          BEQL    30$                     ;IF EQL NO MEMORY AVAILABLE
                  04 A0   51   D1   00D2   337          CMPL    R1,4(R0)                ;FREE BLOCK BIG ENOUGH?
                          F2   1A   00D6   338          BGTRU   10$                     ;IF GTRU NO
                          0E   13   00D8   339          BEQL    20$                     ;IF EQL FREE BLOCK IS EXACT SIZE
                53   51   50   C1   00DA   340          ADDL3   R0,R1,R3                ;CALCULATE ADDRESS OF NEW FREE BLOCK
                     83   80   D0   00DE   341          MOVL    (R0)+,(R3)+             ;COPY LINK TO NEXT FREE BLOCK
                63   60   51   C3   00E1   342          SUBL3   R1,(R0),(R3)            ;CALCULATE SIZE OF NEW FREE BLOCK
                     70   73   DE   00E5   343          MOVAL   -(R3),-(R0)             ;SET LINK TO NEW FREE BLOCK
                     62   60   D0   00E8   344 20$:     MOVL    (R0),(R2)               ;COPY LINK TO NEW FREE BLOCK
                     52   80   9F   00EB   345          MOVAB   (R0)+,R2                ;SET ADR OF ALLOCATED BLOCK, INDICATE SUCCES
                          05   00EE   346 30$:     RSB                             ;
```

```
                                00EF    348          .SBTTL  DEALLOCATE NONPAGED DYNAMIC MEMORY
                                00EF    349 ;+
                                00EF    350 ; EXE$DEANONPAGED - DEALLOCATE NONPAGED DYNAMIC MEMORY
                                00EF    351 ;
                                00EF    352 ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO A NONPAGED POOL.
                                00EF    353 ; IF THE BLOCK IS A SHARED MEMORY BLOCK TYPE, THE BLOCK IS DEALLOCATED TO
                                00EF    354 ; THE SHARED MEMORY POOL.  OTHERWISE, THE BLOCK'S ADDRESS IS CHECKED TO SEE
                                00EF    355 ; IF IT WAS ALLOCATED FROM THE I/O PACKET LOOKASIDE LIST AND IF SO,
                                00EF    356 ; IT IS RETURNED TO THAT LIST.  OTHERWISE IT IS MERGED INTO THE NORMAL
                                00EF    357 ; NONPAGED POOL.
                                00EF    358 ;
                                00EF    359 ; INPUTS:
                                00EF    360 ;
                                00EF    361 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                                00EF    362 ;       IRP$W_SIZE(R0) = SIZE OF BLOCK TO BE DEALLOCATED.
                                00EF    363 ;       IRP$B_TYPE(R0) = TYPE OF BLOCK TO BE DEALLOCATED.
                                00EF    364 ;
                                00EF    365 ; OUTPUTS:
                                00EF    366 ;
                                00EF    367 ;       THE SPECIFIED BLOCK IS RETURNED TO THE APPROPRIATE POOL.
                                00EF    368 ;-
                                00EF    369
                                00EF    370          .ENABL  LSB
                                00EF    371 EXE$DEANONPAGED::                           ;DEALLOCATE NONPAGED DYNAMIC MEMORY
                          48    00EF    372          BSBB    CHECKBLOCK              ;CHECK DEALLOCATION PARAMETERS
                00000000'EF     00F1    373          MOVAB   L^EXE$GL_NONPAGED,R3    ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                          53    00F7
                          83    D5  00F8    374          TSTL    (R3)+                   ;Skip IPL for list
                          1C    E0  00FA    375          BBS     #DSA$V_USER -
                06 0000FE00'EF        00FC    376                  ,DSA$GL_FLAGS, 7$       ;DISABLE ONLY IF IN S/A
                                      0102    377          DSBINT  #^X1F
                                      0108    378 7$:
                          50    D1  0108    379          CMPL    R0,L^EXE$GL_SPLITADR    ;I/O PACKET?
                0000000C'EF         010A
                          1A    1F  010F    380          BLSSU   10$                     ;IF LSSU NO
                                      0111    381 ;+
                                      0111    382 ;       Replaced        INSQUE  (R0),@L^IOC$GL_IRPBL
                                      0111    383 ;-
                00000018'FF   DE    0111    384          MOVAL   @L^IOC$GL_IRPBL,R3      ;WHERE TO INSERT IT
                          53        0117
                          60    63  D0  0118    385          MOVL    (R3),(R0)
                       04 A0    63  DE  011B    386          MOVAL   (R3),4(R0)
                          63    60  DE  011F    387          MOVAL   (R0),(R3)
                          60    DE  0122    388          MOVAL   (R0),L^IOC$GL_IRPBL
                00000018'EF         0124
                          02    11  0129    389          BRB     20$                     ;
                          35    10  012B    390 10$:     BSBB    EXE$DEALLOCATE          ;DEALLOCATE BLOCK
                          50    03  3C  012D    391 20$:     MOVZWL  #RSN$_NPDYNMEM,R0       ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
                          1C    E0  0130    392          BBS     #DSA$V_USER -
                03 0000FE00'EF        0132    393                  ,DSA$GL_FLAGS, 30$      ;ENBINT IF IN S/A
                                      0138    394          ENBINT                          ;ENABLE INTERRUPTS
                                05    0138    395 30$:     RSB                             ;RETURN
                                      013C    396          .DSABL  LSB
```

```
                         013C   398          .SBTTL  CHECK BLOCK PARAMETERS SUBROUTINE
                         013C   399 ;
                         013C   400 ; CHECKBLOCK - CHECK BLOCK PARAMETERS SUBROUTINE
                         013C   401 ;
                         013C   402
                         013C   403 CHECKBLOCK:                                  ;CHECK BLOCK PARAMETERS
     50    0F    D3     013C   404          BITL    #MASK,R0                    ;BLOCK ALIGNED ON BOUNDRY?
           0C    12     013F   405          BNEQ    10$                         ;IF NEQ NO - BAD DEALLOCATION
51     08 A0    3C     0141   406          MOVZWL  IRP$W_SIZE(R0),R1           ;GET SIZE OF BLOCK IN BYTES
     51    0F    C0     0145   407          ADDL    #MASK,R1                    ;ROUND SIZE UP TO NEXT BOUNDRY
     51    0F    CA     0148   408          BICL    #MASK,R1                    ; TRUNCATE SIZE BACK TO MULTIPLE
           14    12     014B   409          BNEQ    20$                         ;IF NEQ OKAY
                        014D   410 10$:     BUG_CHECK BADDALRQSZ                ;BAD DEALLOCATION REQUEST SIZE OR ADDRESS
           8E    D5     015F   411          TSTL    (SP)+                       ;REMOVE RETURN FROM STACK
                 05     C161   412 20$:     RSB                                 ;
```

```
                                0162   414          .SBTTL  GENERAL DEALLOCATION SUBROUTINE
                                0162   415 ;+
                                0162   416 ; EXE$DEALLOCATE - DEALLOCATION SUBROUTINE
                                0162   417 ;
                                0162   418 ; INPUTS:
                                0162   419 ;
                                0162   420 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                                0162   421 ;       R1 = SIZE OF BLOCK IN BYTES
                                0162   422 ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                                0162   423 ;
                                0162   424 ; OUTPUTS:
                                0162   425 ;
                                0162   426 ;       NONE
                                0162   427 ;-
                                0162   428
                                0162   429 EXE$DEALLOCATE::                              ;DEALLOCATE BLOCK
                        52  53  D0  0162   430 10$:    MOVL    R3,R2                   ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
                        53  62  D0  0165   431         MOVL    (R2),R3                 ;GET ADDRESS OF NEXT FREE BLOCK
                            07  13  0168   432         BEQL    20$                     ;IF EQL END OF LIST
                        53  50  D1  016A   433         CMPL    R0,R3                   ;BLOCK LOGICALLY GO HERE?
                            F3  1A  016D   434         BGTRU   10$                     ;IF GTRU NO
                            2D  13  016F   435         BEQLU   50$                     ;IF EQLU DOUBLE DEALLOCATION
                        60  53  D0  0171   436 20$:    MOVL    R3,(R0)                 ;ASSUME NO AGGLOMERATION
                    7E  51  50  C1  0174   437         ADDL3   R0,R1,-(SP)             ;CALCULATE ADDRESS OF END OF BLOCK
                        8E  53  D1  0178   438         CMPL    R3,(SP)+                ;END OF BLOCK EQUAL TO NEXT IN LIST?
                            06  12  017B   439         BNEQ    30$                     ;IF NEQ DO NOT AGGLOMERATE
                        60  83  D0  017D   440         MOVL    (R3)+,(R0)              ;MOVE LINK TO BLOCK BEING RELEASED
                        51  63  C0  0180   441         ADDL    (R3),R1                 ;ACCUMULATE LENGTH OF NEW FREE BLOCK
                            52  DD  0183   442 30$:    PUSHL   R2                      ;CALCULATE ENDING ADDRESS OF PREVIOUS BLOCK
                        82  50  D0  0185   443         MOVL    R0,(R2)+                ;ASSUME NO AGGLOMERATION
                        6E  62  C0  0188   444         ADDL    (R2),(SP)               ;ADD LENGTH TO BLOCK BASE ADDRESS
                        8E  50  D1  018B   445         CMPL    R0,(SP)+                ;END ADDRESS EQUAL TO BLOCK BEING RELEASED?
                            09  12  018E   446         BNEQ    40$                     ;IF NEQ DO NOT AGGLOMERATE BLOCKS
                        51  62  C0  0190   447         ADDL    (R2),R1                 ;ACCUMULATE SIZE OF NEW FREE BLOCK
                        72  60  D0  0193   448         MOVL    (R0),-(R2)              ;MOVE LINK TO PREVIOUS FREE BLOCK
                        50  52  D0  0196   449         MOVL    R2,R0                   ;SET ADDRESS OF NEW FREE BLOCK
                04  A0  51      D0  0199   450 40$:    MOVL    R1,4(R0)                ;SET SIZE OF FREE BLOCK
                            05  019D   451         RSB                             ;
                                019E   452 50$:    BUG_CHECK DOUBLDEALO,FATAL      ;DOUBLE DEALLOCATION OF MEMORY BLOCK
                            00  01B5   453         HALT                            ;HALT FOR NOW
                            E6  11  01B6   454         BRB     50$
```

```
                     01B8    456          .SBTTL  MEMPOOL$INIT                    ;USER MODE MEMORY INITIALIZATION
                000001B8     457 .PSECT  CODE,    SHR,    EXE,    NOWRT,  BYTE
                     01B8    458
                     01B8    459 ;++
                     01B8    460 ;FUNCTIONAL DESCRIPTION:
                     01B8    461 ;
                     01B8    462 ;       This routine allocates how must scratch memory is required and
                     01B8    463 ;       expands the program region to get it. This memory is optionally
                     01B8    464 ;       split between IRP's and other free memory
                     01B8    465 ;
                     01B8    466 ;CALLING SEQUENCING:
                     01B8    467 ;
                     01B8    468 ;       BSBW    USER$MEMINIT
                     01B8    469 ;
                     01B8    470 ;INPUT PARAMETERS:
                     01B8    471 ;
                     01B8    472 ;IMPLICIT INPUTS:
                     01B8    473 ;
                     01B8    474 ;OUTPUT PARAMETERS:     NONE
                     01B8    475 ;
                     01B8    476 ;IMPLICIT OUTPUTS:      NONE
                     01B8    477 ;
                     01B8    478 ;COMPLETION CODES:      NONE
                     01B8    479 ;
                     01B8    480 ;SIDE EFFECTS:          NONE
                     01B8    481 ;--
                     01B8    482
                     01B8    483 MEMPOOL$INIT::
                     01B8    484          PUSHR   #^M<R2,R3>              ; Save working registers
 00000000'8F  DO     01BA    485          MOVL    #SCRIPT$MINCORE,R0      ;SPACE NEEDED FOR SCRIPTS
          50         01C0
 00000000'8F  CO     01C1    486          ADDL2   #QIO$MINCORE,R0         ;PLUS QIO SPACE
          50         01C7
51   50   0A  C7     01C8    487          DIVL3   #10,R0,R1               ;PLUS 10%
     50   51  CO     01CC    488          ADDL2   R1,R0
51   50   05  C5     01CF    489          MULL3   #5,R0,R1                ;IN USER MODE USE 5 TIMES MINIMUM
          1C  EO     01D3    490          BBS     #DSA$V_USER,DSA$GL_FLAGS -
 0000FE00'EF         01D5
          0F         01DA    491                  , 10$                  ;SKIP CALCULATION IN USER MODE
                     01DB    492
 00000000'EF  C3     01DB    493          SUBL3   DS$GA_LASTADR,DS$GL_MEMSIZE -
 00000000'EF         01E1
          51         01E6    494                  , R1                   ;ROOM ABOVE SUPERVISOR
     51   0A  C6     01E7    495          DIVL2   #10,R1                  ;10% OF THAT
                     01EA    496 10$:
51   00   50  F1     01EA    497          ACBL    R0,#0,R1,20$            ;MAX (MIN NECESSARY, AVAILABLE)
               0003  01EE
     50   51  DO     01F0    498          MOVL    R1,R0
50   01FF 8F  A8     01F3    499 20$:     BISW2   #^X1FF,R0               ;PAGE ALIGN
          50  D6     01F8    500          INCL    R0
          1C  EO     01FA    501          BBS     #DSA$V_USER,DSA$GL_FLAGS -
 0000FE00'EF         01FC
          0D         0201    502                  , 30$                  ;USER ALLOCATIONS ARE DIFFERENT
                     0202    503
 00000000'EF  DO     0202    504          MOVL    DS$GA_LASTADR,R1        ;SAVE START ADDRESS OF SPACE
          51         0203
  50   6041  9E     0209    505          MOVAB   (R0)[R1],R0             ;HIGHEST+1 ADDRESS OF SPACE
```

```
                        23   11  020D   506          BRB     40$                     ;JOIN COMMON CODE
                                 020F   507 30$:
                     51 7E   7E  020F   508          MOVAQ   -(SP),R1                ;WHERE VMS WILL RETURN THE ADDRESSES
                  50   F7 8F   78  0212   509          ASHL    #-9,R0,R0               ;MAKE PAGE COUNT
                        50       0216
                                 0217   510          $EXPREG_S R0,(R1)               ;ALLOCATE THE SPACE
                        02   BA  0226   511          POPR    #^M<R1>                 ;GET START ADDRESS
                000001FF 8F   C9  0228   512          BISL3   #^X1FF,(SP)+,R0         ;GET END ADDRESS
                     50   8E       022E
                        50   D6  0230   513          INCL    R0                      ;+1
                                 0232   514 40$:
                                 0232   515 ;+
                                 0232   516 ;       R0      Highest address allocated + 1
                                 0232   517 ;       R1      Lowest address allocated
                                 C232   518 ;-
                        51   D0  0232   519          MovL    R1, L^Ds$GL_LookAside   ; Store start of look-aside            [06]
                00000010'EF      0234
                        50   D0  0239   520          MovL    R0, L^DS$GA_LASTADR     ;PUT BACK NEW AVAILABLE ADDR
                00000000'EF      023B
                        51   D0  0240   521          MOVL    R1,EXE$GL_NONPAGED+4    ;SET ADDRESS OF CHUNK
                00000004'EF      0242
                        61   D4  0247   522          CLRL    (R1)                    ;THIS IS FIRST AND LAST CHUNK
                                 0249   523
                00000014'EF   9E 0249   524          MOVAB   L^IOC$GL_IRPFL,R2       ;ADDR OF LIST HEAD
                        52       024F
                     62 52   D0  0250   525          MOVL    R2,(R2)                 ;INIT LISTHEAD
                  04 A2 52   D0  0253   526          MOVL    R2,4(R2)                ;AND TAIL
                00000000'8F   C3 0257   527          SUBL3   #<SGN$C_IRPCNT*<IRP$C_LENGTH+15>&-16>,R0 -
                     53 50       025D
                                 025F   528                  ,R3                     ;ADDR OF FIRST IRP, LOW LIMIT
                        11   11  025F   529          BRB     60$                     ;COUNT FIRST
                                 0261   530 50$:
                                 0261   531 ;+
                                 0261   532 ;       Replaced        INSQUE (R0),(R2)
                                 0261   533 ;-
                     53 62   D0  0261   534          MOVL    (R2),R3                 ;Address of next
                     60 53   D0  0264   535          MOVL    R3,(R0)                 ;Make flink of new point to next
                  04 A0 52   D0  0267   536          MOVL    R2,4(R0)                ;Make blink of new point to current
                     62 50   D0  026B   537          MOVL    R0,(R2)                 ;Make flink of curr point to new
                  04 A3 50   D0  026E   538          MOVL    R0,4(R3)                ;Make blink of old point to new
                                 0272   539 60$:
                        53   F1  0272   540          ACBL    R3,#-<<IRP$C_LENGTH+15>&-16>,R0 -
                FFFFFFA0 8F      0274
                     FFE5 50     0279
                                 027C   541                  ,50$                    ;DECREMENT ADDR BY LENGTH AND MAKE ONE
                                 027C   542
                     50   51   C3 027C   543          SUBL3   R1,R0,4(R1)             ;SET LENGTH OF REMAINING CHUNK
                        04 A1       027F
                        53   D0  0281   544          MOVL    R3,EXE$GL_SPLITADR      ;SET ADDRESS OF LOWEST IRP
                0000000C'EF      0283
                                 0288   545
                        0C   BA  0288   546          POPR    #^M<R2,R3>              ; Restore registers
                             05  028A   547          RSB                             ;RETURN
                                 028B   548          .END
```

```
$$T1                    = 00000000     D
$ER                     = 00000002     D
$MODULE                   00000000 R   D   03
BUG$CHECK                 ********     X   04
CEB$C_LENGTH            = 00000038     D
CEB$C_SLAVLNG           = 00000044     D
CHECKBLOCK                0000013C R   D   04
DS$GA_LASTADR             ********     X   04
DS$GL_LOOKASIDE           00000010 RG  D   02
DS$GL_MEMSIZE             ********     X   04
DSA$AL_APTMAIL            0000FE00     D
DSA$AT_APTTXT             0000FA00     D
DSA$GL_APTCOM             0000FE04     D
DSA$GL_DEVLEN             0000FE58     D
DSA$GL_ERRNO              0000FE44     D
DSA$GL_EVENT              0000FE48     D
DSA$GL_FLAGS              0000FE00     D
DSA$GL_MSGTYP             0000FE40     D
DSA$GL_PASSES             0000FE08     D
DSA$GL_PASSNO             0000FE54     D
DSA$GL_SECTNO             0000FE10     D
DSA$GL_SID                0000FE14     D
DSA$GL_SUBTNO             0000FE4C     D
DSA$GL_TESTNO             0000FE50     D
DSA$GL_UNITS              0000FE0C     D
DSA$GQ_MSGPTR             0000FE68     D
DSA$GT_DEVNAM             0000FE5C     D
DSA$V_USER              = 0000001C     D
DS_ERRSUP                 ********     X   04
DYN$C_BUFIO             = 00000013     D
DYN$C_CEB               = 00000004     D
DYN$C_IRP               = 0000000A     D
DYN$C_JIB               = 0000002F     D
DYN$C_PCB               = 0000000C     D
DYN$C_PQB               = 0000000D     D
DYN$C_TQE               = 0000000F     D
EXE$ALLOCATE              000000C7 RG  D   04
EXE$ALLOCBUF              00000000 RG  D   04
EXE$ALLOCCEB              00000004 RG  D   04
EXE$ALLOCIRP              00000011 RG  D   04
EXE$ALLOCJIB              00000008 RG  D   04
EXE$ALLOCPCB              00000015 RG  D   04
EXE$ALLOCPQB              0000001D RG  D   04
EXE$ALLOCTQE              00000026 RG  D   04
EXE$ALONONPAGED           00000060 RG  D   04
EXE$DEALLOCATE            00000162 RG  D   04
EXE$DEANONPAGED           000000EF RG  D   04
EXE$GL_NONPAGED           00000000 R   D   02
EXE$GL_SPLITADR           0000000C RG  D   02
IOC$GL_IRPBL            = 00000018 R   D   02
IOC$GL_IRPFL              00000014 RG  D   02
IRP$B_CARCON              00000038     D
IRP$B_EFN                 00000022     D
IRP$B_PRI                 00000023     D
IRP$B_RMOD                0000000B     D
IRP$B_TYPE                0000000A     D
IRP$C_LENGTH              0000005C     D
IRP$K_LENGTH              0000005C     D
IRP$L_ARB                 00000050     D
IRP$L_AST                 00000010     D
IRP$L_ASTPRM              00000014     D
IRP$L_DIAGBUF             00000044     D
IRP$L_EXTEND              0000004C     D
IRP$L_IOQBL               00000004     D
IRP$L_IOQFL               00000000     D
IRP$L_IOSB                00000024     D
IRP$L_IOST1               00000034     D
IRP$L_IOST2               00000038     D
IRP$L_MEDIA               00000034     D
IRP$L_PID                 0000000C     D
IRP$L_SEGVBN              00000040     D
IRP$L_SEQNUM              00000048     D
IRP$L_SVAPTE              0000002C     D
IRP$L_TT_TERM             00000038     D
IRP$L_UCB                 0000001C     D
IRP$L_WIND                00000018     D
IRP$Q_NT_PRVMSK           0000003C     D
IRP$W_ABCNT               0000003C     D
IRP$W_BCNT                00000032     D
IRP$W_BOFF                00000030     D
IRP$W_CHAN                00000028     D
IRP$W_FUNC                00000020     D
IRP$W_OBCNT               0000003E     D
IRP$W_SIZE                00000008     D
IRP$W_STS                 0000002A     D
IRP$W_TT_PRMPT            0000003C     D
JIB$C_LENGTH            = 0000006C     D
MASK                    = 0000000F     D
MEMPOOL$INIT              000001B8 RG  D   04
PCB$B_ASTACT              0000000C     D
PCB$B_ASTEN               0000000D     D
PCB$B_PRI                 0000000B     D
PCB$B_PRIB                0000002F     D
PCB$B_TYPE                0000000A     D
PCB$B_WEFC                0000002E     D
PCB$C_LENGTH              0000008C     D
PCB$K_LENGTH              0000008C     D
PCB$L_ARB                 00000084     D
PCB$L_ASTQBL              00000014     D
PCB$L_ASTQFL              00000010     D
PCB$L_EFC2P               00000058     D
PCB$L_EFC3P               0000005C     D
PCB$L_EFCS                00000050     D
PCB$L_EFCU                00000054     D
PCB$L_EFWM                0000004C     D
PCB$L_JIB                 00000078     D
PCB$L_OWNER               0000001C     D
PCB$L_PHD                 00000064     D
PCB$L_PHYPCB              00000018     D
PCB$L_PID                 00000060     D
PCB$L_PQB                 0000004C     D
PCB$L_SQBL                00000004     D
PCB$L_SQFL                00000000     D
PCB$L_STS                 00000024     D
```

```
PCB$L_UIC                       00000088    D
PCB$L_WSSWP                     00000020    D
PCB$L_WTIME                     00000028    D
PCB$Q_PRIV                      0000007C    D
PCB$T_LNAME                     00000068    D
PCB$T_TERMINAL                  00000044    D
PCB$W_APTCNT                    00000030    D
PCB$W_ASTCNT                    00000038    D
PCB$W_BIOCNT                    0000003A    D
PCB$W_BIOLM                     0000003C    D
PCB$W_DIOCNT                    0000003E    D
PCB$W_DIOLM                     00000040    D
PCB$W_GPGCNT                    00000034    D
PCB$W_GRP                       0000008A    D
PCB$W_MEM                       0000C088    D
PCB$W_MTXCNT                    0000000E    D
PCB$W_PPGCNT                    00000036    D
PCB$W_PRCCNT                    00000042    D
PCB$W_SIZE                      00000008    D
PCB$W_STATE                     0000002C    D
PCB$W_TMBU                      00000032    D
PQB$C_LENGTH                  = 000008C8    D
PR$_IPL                       = 00000012    D
QIO$MINCORE                     ********  X   04
RSN$_NPDYNMEM                 = 00000003    D
SCRIPT$MINCORE                  ********  X   04
SGN$C_IRPCNT                    ********  X   04
SIZ...                        = 00000001    D
SS$_INSFMEM                   = 00000124    D
SS$_NORMAL                    = 00000001    D
SYS$EXPREG                      ********  GX  04
T_CGF                           00000007 R  D   03
```

+-----------------+
! Psect synopsis !
+-----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000001C (28.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	0000001C (28.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	0000028B (651.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T1	=00000000	510 (2)	510 (2)
$ER	=00000002	247 (2)	#-247 (2)
$MODULE	00000000-R	131 (2)	247 (2)
BUG$CHECK	00000000-XR		305 (2) 410 (2) 452 (2)
CEB$C_LENGTH	=00000038		215 (2)
CEB$C_SLAVLNG	=00000044		216 (2)
CHECKBLOCK	0000013C-R	403 (2)	#-372 (2)
DS$GA_LASTADR	00000C00-XR		#-493 (2) #-504 (2) #-520 (2)
DS$GL_LOOKASIDE	00000010-R	117 (2)	#-519 (2)
DS$GL_MEMSIZE	00000000-XR		#-493 (2)
DSA$GL_FLAGS	0000FE00		280 (2) 301 (2) 376 (2) 393 (2) 490 (2) 501 (2)
DSA$V_USER	=0000001C		#-279 (2) #-300 (2) #-375 (2) #-392 (2) #-490 (2) #-501 (2)
DS_ERRSUP	00000000-XR		247 (2)
DYN$C_BUFIO	=00000013		#-211 (2)
DYN$C_CEB	=00000004		#-214 (2)
DYN$C_IRP	=0000000A		#-223 (2)
DYN$C_JIB	=0000002F		#-219 (2)
DYN$C_PCB	=0000000C		#-226 (2)
DYN$C_PQB	=0000000D		#-230 (2)
DYN$C_TQE	=0000000F		#-234 (2)
EXE$ALLOCATE	000000C7-R	332 (2)	#-298 (2)
EXE$ALLOCBUF	00000000-R	210 (2)	
EXE$ALLOCCEB	00000004-R	213 (2)	
EXE$ALLOCIRP	00000011-R	222 (2)	
EXE$ALLOCJIB	00000008-R	218 (2)	
EXE$ALLOCPCB	00000015-R	225 (2)	
EXE$ALLOCPQB	0000001D-R	229 (2)	
EXE$ALLOCTQE	00000026-R	233 (2)	
EXE$ALONONPAGED	00000060-R	275 (2)	#-238 (2)
EXE$DEALLOCATE	00000162-R	429 (2)	#-390 (2)
EXE$DEANONPAGED	000000EF-R	371 (2)	
EXE$GL_NONPAGED	00000000-R	110 (2)	296 (2) 373 (2) #-521 (2)
EXE$GL_SPLITADR	0000000C-R	115 (2)	#-379 (2) #-544 (2)
IOC$GL_IRPBL	=00000018-R	127 (2)	384 (2) #-388 (2)
IOC$GL_IRPFL	00000014-R	125 (2)	126 (2) 127 (2) 288 (2) 524 (2)
IRP$B_TYPE	0000000A		#-242 (2)
IRP$C_LENGTH	0000005C		215 (2) 216 (2) #-235 (2) #-283 (2) #-527 (2) #-540 (2)
IRP$W_SIZE	00000008		#-241 (2) #-406 (2)
JIB$C_LENGTH	=0000006C		#-220 (2)
MASK	=0000000F	102 (2)	#-235 (2) #-276 (2) #-277 (2) #-283 (2) #-404 (2) #-407 (2) #-408 (2)
MEMPOOL$INIT	000001B8-R	483 (2)	
PCB$C_LENGTH	0000008C		#-227 (2)
PQB$C_LENGTH	=000008C8		#-231 (2)
PR$_IPL	=00000012		#-281 (2) #-302 (2) #-377 (2) #-394 (2)
QIO$MINCORE	00000000-XR		#-486 (2)
RSN$_NPDYNMEM	=00000003		#-391 (2)

```
SCRIPT$MINCORE                  00000000-XR                     #-485     (2)
SGN$C_IRPCNT                    00000000-XR                     #-527     (2)
SS$_INSFMEM                     =00000124                       #-246     (2)
SS$_NORMAL                      =00000001                       #-294     (2)
SYS$EXPREG                      00000000-XR                       510     (2)
T_CGF                           00000007-R    132     (2)         247     (2)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ASNPUSH	1	510 (2)	510 (2)
$CEBDEF	2	82 (2)	82 (2)
$DEFINI	1	81 (2)	81 (2) 82 (2) 83 (2) 84 (2) 85 (2) 86 (2) 87 (2) 88 (2) 89 (2) 90 (2) 91 (2) 92 (2) 93 (2) 94 (2)
$DS_DSADEF	5	81 (2)	81 (2)
$DYNDEF	2	83 (2)	83 (2)
$EXPREG_S	1	510 (2)	510 (2)
$IPLDEF	1	84 (2)	84 (2)
$IRPDEF	4	85 (2)	85 (2)
$JIBDEF	3	86 (2)	86 (2)
$PCBDEF	4	87 (2)	87 (2)
$PQBDEF	3	88 (2)	88 (2)
$PRDEF	4	89 (2)	89 (2)
$PUSHADR	1	247 (2)	247 (2) 510 (2)
$RSNDEF	1	90 (2)	90 (2)
$SHBDEF	1	91 (2)	91 (2)
$SHDDEF	3	92 (2)	92 (2)
$SSDEF	21	93 (2)	93 (2)
$TQEDEF	2	94 (2)	94 (2)
ASSUME	1	215 (2)	215 (2) 216 (2)
BUG_CHECK	1	305 (2)	305 (2) 410 (2) 452 (2)
DSBINT	1	281 (2)	281 (2) 377 (2)
ENBINT	1	302 (2)	302 (2) 394 (2)
ERRSUP_S	1	247 (2)	247 (2)
MODNAM	1	131 (2)	131 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.10	00:00:00.26
Command processing	140	00:00:00.76	00:00:01.91
Pass 1	878	00:00:15.60	00:00:20.33
Symbol table sort	0	00:00:01.55	00:00:01.73
Pass 2	169	00:00:02.72	00:00:05.52
Symbol table output	19	00:00:00.15	00:00:00.15
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	34	00:00:00.29	00:00:00.41
Assembler run totals	1283	00:00:21.21	00:00:30.35

The working set limit was 1000 pages.
84638 bytes (166 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1039 non-local and 28 local symbols.
548 source lines were read in Pass 1, producing 0 object records in Pass 2.
113 pages of virtual memory were used to define 33 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	5
DRB1:[DS.WORK]DS.MLB;218	3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	9
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	28

1284 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) MEMALC/UPDA=(MEMALC.UPD,MEMALC.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 155 Libraries, Macros, and External Symbols
(1) 210 Own Storage
(1) 245 Data Storage
(1) 254 MapMem Routine - Map all of memory
(1) 460 MAPMEM$IOSPACE Setup page tables for I/O space
(3) 519 ACCESSABLE check a page for accessability
(4) 569 STACKPROT
(5) 625 MAP FREE MEMORY PROCEDURE.
(7) 717 GET A BLOCK OF VIRTUAL MEMORY.
(10) 867 RELEASE A BLOCK OF VIRTUAL MEMORY.
(13) 1005 TURN MEMORY MANAGEMENT ON.
(14) 1054 TURN MEMORY MANAGEMENT OFF.
(15) 1106 DSR$MMENABLE Enable to disable memory management
(16) 1147 SET PROTECTION ON PAGES.
(19) 1307 CALCULATE PTE ADDRESS SUBROUTINE
(21) 1391 DSX$MAPDBGBLOCK - MAP A BLOCK OF MEMORY FOR THE DEBUGGER [23]
(22) 1432 DSX$FREEDBGSYM - FREE MEMORY MAPPED TO DEBUGGER SYMTABS
(23) 1462 MAP_DEBUGGER - MAP THE DEBUGGER'S MEMORY SPACE [23]
(24) 1487 UNMAP_DEBUGGER - UNMAP THE DEBUGGER'S MEMORY SPACE [23]

```
0000      1          .TITLE  MEMMGT  *** MEMMGT Memory management setup/control
0000      2          .IDENT  /06-29/
0000      3          .NoShow Conditionals
0000      4          .DSABL  GBL
0000      5
0000      6 ;
0000      7 ; COPYRIGHT (C) 1977, 1981, 1983
0000      8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000      9 ;
0000     10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000     13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000     14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
C000     15 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000     16 ; REMAIN IN DEC.
0000     17 ;
0000     18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000     19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000     20 ; CORPORATION.
0000     21 ;
0000     22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000     23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     24
0000     25 ;++
0000     26 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     27 ;
0000     28 ; ABSTRACT:
0000     29 ;
0000     30 ; ENVIRONMENT:
0000     31 ;
0000     32 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000     33 ;
0000     34 ; MODIFIED BY:
0000     35 ;                KEN CHAPMAN     30-NOV-77       VERSION 02
0000     36 ;       01       DSPR #63 - FIXED $DS_GETBUF_L OVERLAYING HARDWARE P-TABLES.
0000     37 ;
0000     38 ;                NICK HOWGATE    09-JAN-78       VERSION 03 (ESSAA-3.06)
0000     39 ;       02       DSPR #41 - TURN ON MEMORY MANAGEMENT FIX.
0000     40 ;       03       ADDED $SETPRT ROUTINE.
0000     41 ;
0000     42 ;                KEN CHAPMAN     26-JAN-78       VERSION 04 (ESSAA-3.07)
0000     43 ;       04       MORE FIXES TO MEMORY MANAGEMENT PAGE TABLES.
0000     44 ;
0000     45 ;                KEN CHAPMAN     06-APR-78       VERSION 05 (ESSAA-4.01)
0000     46 ;       05       ADDED P1 AND SYSTEM BUFFER ALLOCATION CAPABILITY.
0000     47 ;                Roger Riggs     12-Jun-78       Version 06 (ESSAA-4.03)
0000     48 ;       06       Fixed Release of P1 or SYS space conflict of PTE's
0000     49 ;       07       MAPMEM cleanup and set DS$GL_MEMSIZE to bytes of physical
0000     50 ;                memory.
0000     51 ;                N. HOWGATE      14-NOV-78       VERSION 07 (ESSAA-5.01)
0000     52 ;       08       CLEAR MACHINE CHECKS CORRECTLY
0000     53 ;       09       FIX BUG IN $DS_SETPRT
0000     54 ;       10       Removed dependence of MAPFREE on L$L_UNITS
0000     55 ;
0000     56 ;       11       Added processor dependent I/O space mapping
0000     57 ;                Roger Riggs     6-Sept-1979
```

```
                     58 ;        12      Added IO$GQ_PHYSICAL, used to determine where to map
                     59 ;                I/O space in P1 space.
                     60 ;                Roger Riggs, 24-Jan-1980,  Version 5.2
                     61 ;        12      Re-arranged system page table to allow system space addresses
                     62 ;                to map 1 for 1 P0 addresses in the range %X10000 to the
                     63 ;                top of the P0 page tables.  References to this range will
                     64 ;                reference the same addresses with or without memory management
                     65 ;                enabled.
                     66 ;                Roger Riggs, 5-Mar-1980, Version 5.3
                     67 ;        13      Made page at APT$AT_APTTXT valid, affects APT when
                     68 ;                Memory Management is on.
                     69 ;                Roger Riggs, 12-Mar-1980, Version 5.3
                     70 ;        14      Added byte to save state of memory managment, allows
                     71 ;                Memory managment to be enabled or disabled after cleanup
                     72 ;                to restore it to what the operator indicated.
                     73 ;                Also if the operator enabled memory managment then the program
                     74 ;                is not allowed to turn it off.  DS$GB_MM_ENB is set
                     75 ;                by CLI command SET MM ON and SET MM OFF.
                     76 ;                Also added routines and support for XDELTA
                     77 ;                Roger Riggs, 21-May-1980, Version 5.4
                     78 ;        15      Removed upper limit on amount of memory checked
                     79 ;                Fix DS$EXPREG to set the protection of pages allocated to UW.
                     80 ;                In DS$MMON AND DS$MMOFF return the previous state of MM as
                     81 ;                SS$_WASSET OR SS$_WASCLR and return DS$_WARNING from
                     82 ;                DS$MMOFF if operator set MMON and program is disallowed from
                     83 ;                turning it off, Add code to flush the translation buffer
                     84 ;                before turning MM ON, Set protections on stacks so
                     85 ;                underflows will can be caught by MM.
                     86 ;
                     87 ;                Dave Butenhof   17-jul-1980
                     88 ;        16      Call MAPFREE from MAPMEM to free up memory; to allow
                     89 ;                GETBUF from Supervisor before program is loaded.
                     90 ;                Specifically, for HELP utility.
                     91 ;
                     92 ;                Roger Riggs, 17-Sep-1980, Version 6.0
                     93 ;        17      Changed MAPFREF to check program loaded flag to determine
                     94 ;                how much of low memory to map.
                     95 ;
                     96 ;                Dave Butenhof, 23-feb-1981, version 6.3
                     97 ;        18      Change a word-relative to longword-relative.
                     98 ;        19      Fix two bugs: in EXE$SETPRT, correct return address array
                     99 ;                start address--BICL3 first two operands were reversed.
                    100 ;                in EXE$CNTREG, Comparison of P0 and P1/SYS PTE resulted in
                    101 ;                ERRSUP if M bit was altered, and ignored V and PFN; fix the
                    102 ;                compare mask.
                    103 ;
                    104 ;                - Dave Butenhof, 08-Jun-1981, version 6.4
                    105 ;        20      Fix SETPRT some more.  Do a CHMK to force cpu
                    106 ;                to KERNEL mode, so we can use MFPR.
                    107 ;
                    108 ;        21      - dave butenhof, 3-sep-1981, version 6.5
                    109 ;                alter to use DS with new XDELTA
                    110 ;
                    111 ;        22      Jack Stansbury, Version 6.8, May 28, 1982
                    112 ;                Add an entry point to the ExpReg routine for use by CRD.
                    113 ;                In stand-alone, ExpReg was allocating space beginning at 200,
                    114 ;                which is a no-no! Fix it to expand above the Supervisor (as it
```

```
0000   115 ;                   does on-line).
0000   116 ;
0000   117 ;           23      Richard Brown, Version 6.8, 25-June-82
0000   118 ;                   Addition of routines which map a block
0000   119 ;                   of memory for the debugger to use.
0000   120 ;
0000   121 ;           24      Richard Brown, Version 6.9, 4-August-82
0000   122 ;                   Addition of routine to deallocate memory assigned
0000   123 ;                   to the debugger's symbol tables.
0000   124 ;
0000   125 ;           25      Bob Bergazzi    5-Aug-82        Version 6.9
0000   126 ;                   Changed MAPFREE to call DSR$CheckLoad (SHOWMEM)
0000   127 ;                   to see if a program is loaded - fixes "SET MM ON,
0000   128 ;                   ATTACH, Translation not valid fault" bug.
C000   129 ;
0000   130 ;           26      Jack Stansbury  25-Aug-1982     Version 6.9
0000   131 ;                   Added two other vaiables: DS$GL_Actual_MemSize and
0000   132 ;                   DS$GL_Set_MemSize. The former is the actual size of the memory
0000   133 ;                   of the machine. The latter is the size that the operator wishes
0000   134 ;                   to set the memory. Also added .Library statements, and changed
0000   135 ;                   the PSect from SEP to what they should be. Also changed the
0000   136 ;                   insufficient memory message from a PrintF to a Print.
0000   137 ;
0000   138 ;           27      Jack Stansbury  25-Aug-1982     Version 6.9
0000   139 ;                   Fixed several truncation errors.
0000   140 ;
0000   141 ;           28      Bob Bergazzi    2-Nov-1982      Version 6.9
0000   142 ;                   Changed the ACCESSABLE routine so that it only copies the
0000   143 ;                   first quadword of a page rather than every quadword on the
0000   144 ;                   page. This results in considerable savings of startup time,
0000   145 ;                   especially for NEBULA systems with large amounts of memory
0000   146 ;                   (greater than 3 Meg took longer than 10 seconds, causing
0000   147 ;                   APT to consider the supervisor hung).
0000   148 ;
0000   149 ;           29      Bob Bergazzi    29-Aug-1983     Version 6.13
0000   150 ;                   Changed the value of the minimum memory size required from
0000   151 ;                   256K to 512K bytes.
0000   152 ;
0000   153 ;--
```

```
                     0000   155          .SubTitle       Libraries, Macros, and External Symbols
                     0000   156
                     0000   157 ;
                     0000   158 ; INCLUDE FILES:
                     0000   159 ;
                     0000   160
                     0000   161          .Library        /Sys$Library:Lib/                                    ;[26
                     0000   162          .Library        /$DS/                                               ;[26
                     0000   163          .Library        /$Diag/                                             ;[26
                     0000   164
                     0000   165 ;
                     0000   166 ; EQUATED SYMBOLS:
                     0000   167 ;
                     0000   168
                     0000   169          CMKDEF
                     0000   170          DSFDEF
                     0000   171
                     0000   172          $CNTREGDEF
                     0000   173          $DS_BITDEF
                     0000   174          $DS_CFDEF
                     0000   175          $DS_DSADEF
                     0000   176          $DS_DSDEF
                     0000   177          $DS_ENVDEF
                     0000   178          $DS_GETBUF_DEF
                     0000   179          $DS_GETMEM_DEF
                     0000   180          $DS_HDRDEF
                     0000   181          $DS_RELBUF_DEF
                     0000   182          $DS_RELMEM_DEF
                     0000   183          $DS_SCBDEF
                     0000   184          $DS_TypeDef                     ; The typecodes                     [26]
                     0000   185
                     0000   186          $EXPREGDEF
                     0000   187          $IRPDEF
                     0000   188          $PHDDEF
                     0000   189          $PRDEF
                     0000   190          $PRTDEF
                     0000   191          $PSLDEF
                     0000   192          $PTEDEF
                     0000   193            $VIELD  PTE,<29-9>,<<10,,M>>
                     0000   194          $SETPRTDEF
                     0000   195          $SSDEF
                     0000   196
0000007E             0000   197 K_P1PAGES=126                          ; Number of P1 pages to create
                     0000   198
                     0000   199          .EXTRN  POPT_BASE,      MAP$IOSPACE,    MEMPOOL$INIT
                     0000   200          .EXTRN  MMG$GL_SPTBASE, DS$AQ_SYSSRV,   DS$GA_LASTADR
                     0000   201          .EXTRN  SYS$UNWIND,     DS$GL_BUFCNT,   DSR$SETIMR_INI
                     0000   202          .EXTRN  DS$GQ_PHYADR,   DS_ERRSUP,      DSX$Print                   ;[26
                     0000   203          .EXTRN  IO$GQ_PHYSICAL
                     0000   204          .EXTRN  SCB_UNKINT,STACK_BASE,ISTKPTR,KSTKPTR,ESTKPTR,SSTKPTR,USTKPTR
                     0000   205          .EXTRN  DS$R_PRGSIZ,    DS$A_PRGBGN,    DS$GL_FLAGS
                     0000   206          .EXTRN  DEBUG_LO,       DEBUG_HI                                    ;[23
                     0000   207          .EXTRN  SYMTAB_HI                                                   ;[24
                     0000   208          .EXTRN  DSR$CheckLoad                                               ;[25
```

```
                        0000   210          .Subtitle       Own Storage
                        0000   211          .PSect          Work, NoShr, NoExe, Wrt, Long                          ;[26
                        0000   212
                        0000   213 ;
                        0000   214 ; OWN STORAGE:
                        0000   215 ;
                        0000   216
00000004                0000   217 L_POLEN:          .BLKL   1       ; P0 (PHYSICAL MEMORY) LENGTH.
00000008                0004   218 A_P1BUFPTE:       .BLKL   1       ; P0 ADDRESS OF P1 BUFFER PTE'S.
0000000C                0008   219 A_P1BUFVA:        .BLKL   1       ; P1 BUFFER VIRTUAL ADDRESS.
00000010                000C   220 A_SPTEND:         .BLKL   1       ; SYS PAGE TABLE END.
00000014                0010   221 DS$GL_MEMSIZE::   .BLKL   1       ; BYTES OF PHYSICAL MEMORY
                        0014   222
                        0014   223 DS$GL_Actual_MemSize::           ; The actual size of memory                    [26]
00000018                C014   224                   .BlkL   1       ;                                              [26]
                        0018   225
                        0018   226 DS$GL_Set_MemSize::              ; The 'SET' memory size                        [26]
00000000                0018   227                   .Long   0       ; Initialized to zero                          [26]
                        001C   228
0000001D                001C   229 DS$GB_MM_ENB::     .BLKB   1       ; 0 if MM OFF, 1 if ON
                        001D   230
                        001D   231 ;
                        001D   232 ; Symbols used to give initial values to X6 thru XD.
                        001D   233 ;
                        001D   234
00000000                001D   235 PFN$AW_SWPVBN    == L_POLEN              ; X6                                   [21]
00000004                001D   236 PFN$AL_PTE       == A_P1BUFPTE           ; X7
00000008                001D   237 PFN$AL_BAK       == A_P1BUFVA            ; X8
0000000C                001D   238 PFN$AW_REFCNT    == A_SPTEND             ; X9
00000000                001D   239 PFN$AW_FLINK     == 0                    ; XA                                   [21]
00000000                001D   240 PFN$AW_BLINK     == 0                    ; XB                                   [21]
00000010                001D   241 PFN$AB_STATE     == DS$GL_MEMSIZE        ; XC Bytes physical memory
00000000                001D   242 PFN$AB_TYPE      == 0                    ; XD                                   [21]
00000000                001D   243 XDS$INIT         == 0                    ; Address of command string NOP
```

```
                                    001D   245          .Subtitle       Data Storage
                                00000000   246          .PSect          Data, Shr, NoExe, NoWrt, Long
                                    0000   247
                                    0000   248          MODNAM  MEMMGT
                                    0007   249
                                    0007   250 T_INSFMEM:
4C 55 21 20 79 6C 6E 4F 20 3F 3F 00'0007   251          .ASCIC  "?? Only !UL pages memory present, 1024 are required.!/"        ;E29
72 6F 6D 65 6D 20 73 65 67 61 70 20 0013
31 20 2C 74 6E 65 73 65 72 70 20 79 001F
75 71 65 72 20 65 72 61 20 34 32 30 002B
                  2F 21 2E 64 65 72 69 0037
                              36    0007
                                    003E   252
```

```
                        003E    254          .SBTTL  MapMem Routine - Map all of memory
                    00000000    255          .PSect  Code, Shr, Exe, NoWrt, Long
                        0000    256
                        0000    257 ;++
                        0000    258 ; FUNCTIONAL DESCRIPTION:
                        0000    259 ;
                        0000    260 ;       MAPS AVAILABLE MEMORY AND SETS UP MEMORY MANAGEMENT PAGE TABLES.
                        0000    261 ;
                        0000    262 ; Note:
                        0000    263 ;       ESKAZ (memory management testing) expects the length register
                        0000    264 ;       for each addressing space to exclude 1 valid PTE. As a result
                        0000    265 ;       there is a more valid PTE just beyond the length register.
                        0000    266 ;
                        0000    267 ; Note:
                        C000    268 ;       This allows a value to be inserted in the longword DS$GL_Set_MemSize
                        0000    269 ;       that will be used as the memory size, rather than the actual memory
                        0000    270 ;       size. This way, the operator can, for example, set up the memory for
                        0000    271 ;       a 256K machine in a machine that actually has 2M (or whatever). Note
                        0000    272 ;       that this user-set memory size must fulfill the following two
                        0000    273 ;       conditions:
                        0000    274 ;               1. Must be greater than 0. 0 is the default.
                        0000    275 ;               2. Must be less than or equal to the actual memory size.
                        0000    276 ;
                        0000    277 ; CALLING SEQUENCE:
                        0000    278 ;
                        0000    279 ;       BSBW    MAPMEM
                        0000    280 ;
                        0000    281 ; INPUT PARAMETERS:     NONE
                        0000    282 ;
                        0000    283 ; IMPLICIT INPUTS:      NONE
                        0000    284 ;
                        0000    285 ; OUTPUT PARAMETERS:    NONE
                        0000    286 ;
                        0000    287 ; IMPLICIT OUTPUTS:     NONE
                        0000    288 ;
                        0000    289 ; COMPLETION CODES:     NONE
                        0000    290 ;
                        0000    291 ; SIDE EFFECTS:
                        0000    292 ;
                        0000    293 ;       Plenty!
                        0000    294 ;--
```

```
                                0000   296 MAPMEM::
                                0000   297
                                0000   298 ;+
                                0000   299 ; Determine how much memory is present and create page table for it.
                                0000   300 ; The last page/PTE is outside the length register and reserved for
                                0000   301 ; ESKAZ.
                                0000   302 ;-
                     007C 8F    BB  0000   303          PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save all the registers that are used  [26]
          55  00000000'EF       DE  0004   304          MOVAL   P0PT_BASE,R5            ; Base Page Table
                        53      D4  000B   305          CLRL    R3                      ; Start with PFN = 0
                                000D   306
                  6543   6543    C2  000D   307 10$:     SUBL2   (R5)[R3],(R5)[R3]       ; CLEAR PTE
                7E   53   09    78  0012   308          ASHL    #9,R3,-(SP)             ; MAKE ADDRESS OF FIRST LOC
             00000207'EF   01    FB  0016   309          CALLS   #1,L^ACCESSABLE         ; Can the page be read?                 [27]
                     10 50      E9  001D   310          BLBC    R0,20$                  ; Branch if Not accessable
           6543   A1800000 E3    9E  0020   311          MOVAB   PTE$M_VALID ! PTE$M_OWN ! PTE$C_UW(R3), -
                                0028   312                  (R5)[R3]                ; Make page valid
        DD 53   000FFFFF 8F    F2  0028   313          AOBLSS  #^XFFFFF,R3,10$         ; Repeat until aligned
                                0030   314
                                0030   315          ;+
                                0030   316          ; R3 now equals the number of pages in memory (i.e., number of K Bytes  [26]
                                0030   317          ; equal to R3 / 2).                                                     [26]
                                0030   318          ;-
                                0030   319
     00000014'EF   53   09   78  0030   320 20$:     AshL    #9, R3, -               ; Use the value in R3 (^9) as the       [26]
                                0038   321                    DS$GL_Actual_MemSize  ; ... actual memory size                [26]
          56  00000018'EF      D0  0038   322          MovL    DS$GL_Set_MemSize, R6   ; Get the user-set memory size (pages)  [26]
                        08      13  003F   323          Beql    22$                     ; If = 0, then branch                   [26]
                     53   56    D1  0041   324          CmpL    R6, R3                  ; Compare user-set and actual           [26]
                        03      14  0044   325          Bgtr    22$                     ; If user-set > actual, use actual      [26]
                                0046   326
                                0046   327          ;+
                                0046   328          ; Now the user-set memory size (in R6) satisfies the following:         [26]
                                0046   329          ;       0 < R6 <= R3                                                    [26]
                                0046   330          ; Remember that both of these quanities are number of pages.           [26]
                                0046   331          ;-
                                0046   332
                     53   56    D0  0046   333          MovL    R6, R3                  ; Use the user-set - not the actual     [26]
                                0049   334
     00000010'EF   53   09   78  0049   335 22$:     ASHL    #9,R3,DS$GL_MEMSIZE     ; Set memory size                       [26]
          00000400 8F   53    D1  0051   336          CMPL    R3,#1024                ; 512K memory present?                  [26]
                        15      18  0058   337          BGEQ    25$                     ; Branch if enough memory
                                005A   338
                                005A   339          $Print  -                       ; Print an error message                [26]
                                005A   340                    #DS$K_Type_General_Error, -; ... a general error        [26]
                                005A   341                    #DS$K_PrintF, -         ; ... use PrintF                        [26]
                                005A   342                    T_InstMem, -            ; ... the error message                 [26]
                                005A   343                    R3                      ; ... the number of pages               [26]
                                006F   344
          65   78000000 8F    CA  006F   345 25$:     BICL    #PTE$M_PROT,(R5)        ; No access to page zero
               50   53   01    C3  0076   346          SUBL3   #1,R3,R0                ; Reserve last 1 PTE'S for ESKAZ
          00000000'EF   50    D0  007A   347          MOVL    R0,L_P0LEN              ; Length in longwords of P0PT
                        09      50  0081   348          MTPR    R0, #PR$_P0LR           ; SET UP P0 LENGTH REGISTER.
                                0084   349
                     53   7F 8F    93  0084   350 30$:     BITB    #^X7F,R3                ; Offset page aligned
                                0C      13  0088   351          BEQL    40$             ; Branch if so
                      6543   63    9E  008A   352          MOVAB   PTE$C_NA(R3),(R5)[R3]   ; Make invalid page
```

```
                EE 53   000FFFFF 8F    F2  008E   353          AOBLSS  #^XFFFFF,R3,30$         ; Repeat until aligned
                                                0096   354
                     55     6543      DE  0096   355 40$:     MOVAL   (R5)[R3],R5             ; Point to SPT base
```

```
                                009A   357
                                009A   358 ;+
                                009A   359 ; Setup system page tables
                                009A   360 ; Map the normal system service vectors to their supervisor counterparts
                                009A   361 ; but do not allow access to catch programming errors.
                                009A   362 ; Map pages from 80000800 to 80010000 N/A and invalid, available for expreg
                                009A   363 ; in system space.
                                009A   364 ; Map pages from 80010000 to 80000000+end of P0 Page table user read, kernel
                                009A   365 ; writeable. Makes system space map to physical memory.
                                009A   366 ; Map each page allocated to the P0 page tables and set the P0 Base and
                                009A   367 ; Length registers.
                                009A   368 ; Allocate 16 System PTEs for EXPREG to use in allocating SYSTEM space.
                                009A   369 ; Allocate 32 PTEs to map P1 page tables and set the P1 base and length
                                009A   370 ; registers. Also allocate 1 PTE for the space allocated by EXPREG in P1 space.
                                C09A   371 ; And reserve two PTEs, one inside the length register and one outside for
                                009A   372 ; ESKAZ.
                                009A   373 ;-
                                009A   374
                                009A   375
                                009A   376 50$:
             0C    55    DA   009A   377          MTPR    R5, #PR$_SBR                ; SET SYSTEM BASE REGISTER.
 00000000'EF    55    D0   009D   378          MOVL    R5, MMG$GL_SPTBASE          ; Save pointer to system page table
             53    D4   00A4   379          CLRL    R3                          ; Start with PFN 0 in system space
     51  00'8F    9A   00A6   380          MOVZBL  #DS$AQ_SYSSRV @-9, R1       ; SYSTEM SERVICE PAGE NUMBER.
                           00AA   381
   6543  81800000 E1    9E   00AA   382 60$:     MOVAB   PTE$M_VALID ! PTE$C_NA ! PTE$M_OWN(R1), -
                           00B2   383                  (R5)[R3]                    ; Set no access to  virtural 80000000
             51    D6   00B2   384          INCL    R1                          ; Next page
          F2 53    04   F2   00B4   385          AOBLSS  #4,R3,60$                   ; Fill 4 pages
                           00B8   386
               6543    D4   00B8   387 65$:     CLRL    (R5)[R3]                    ; Clear next PTE
    F5 53   00000080 8F   F2   00BB   388          AOBLSS  #128,R3,65$                 ; Fill rest of page with avail SPTE's
                           00C3   389
51  00000010'EF    F7 8F   78   00C3   390          ASHL    #-9,DS$GL_MEMSIZE,R1        ; Get PFN of last page + 1
   6543   F1800000 E3   9E   00CC   391 70$:     MOVAB   PTE$M_VALID ! PTE$C_URKW ! PTE$M_OWN (R3), -
                           00D4   392                  (R5)[R3]                    ; Map 1 for 1 valid and user readable
          F4 53    51   F2   00D4   393          AOBLSS  R1,R3,70$                   ; Map up to end of P0 PT
    08   80000000'8F    DA   00D8   394          MTPR    #^X80000000 ! P0PT_BASE, -
                           00DF   395                  #PR$_P0BR                   ; Set P0 Base register
```

```
                                00ODF   397
                                00ODF   398 ;+
                                00ODF   399 ; Setup PTE's to map P1 page tables
                                00ODF   400 ;-
                                00ODF   401
            50   53    09   78 00ODF   402         ASHL    #9,R3,R0                ; SVA for this PFN
       50    80000000 8F   C8 00E3   403         BISL    #^X80000000,R0          ; Make SVA of first P1 PTE
                                00EA   404
 54 00000000'EF   15    09   EF 00EA   405         EXTZV   #9,#21,IO$GQ_PHYSICAL,R4; Get PFN of lowest I/O space
                                00F3   406
       54    0000007F 8F   C2 00F3   407         SUBL    #127,R4                 ; PFN of scratch area
             0B    54   DA 00FA   408         MTPR    R4,#PR$_P1LR            ; Set length register
             54    D7   00FD   409         DECL    R4                      ; Base is really 128 pages away
                                00FF   410
          51   54    04   C5 00FF   411         MULL3   #4,R4,R1                ; Make byte offset
             50   51    C2 0103   412         SUBL    R1,R0                   ; byte address of P1 base
             0A   50    DA 0106   413         MTPR    R0, #PR$_P1BR           ; Set P1 base register
                                0109   414
       52    01800000 8F   D0 0109   415         MOVL    #PTE$M_OWN,R2           ; Initial PTE protection
                                0110   416
          6543   52    D0 0110   417         MOVL    R2,(R5)[R3]             ; Free SPTE for P1 scratch
             53    D6   0114   418         INCL    R3                      ; Update free PTE pointer
                                0116   419
       50    00000000'EF   7D 0116   420         MOVQ    IO$GQ_PHYSICAL,R0       ; Get I/O space address range
             51    50    C2 011D   421         SUBL    R0,R1                   ; Length of I/O space
    50   51    0E    10   EF 0120   422         EXTZV   #16,#14,R1,R0           ; Get SPTE's required
         50    02 A340   9E 0125   423         MOVAB   2(R3)[R0],R0            ; Final PTE + 2 for ESKAZ
                                012A   424
          6543   52    D0 012A   425 80$:      MOVL    R2,(R5)[R3]             ; Set PTE protection
       F8 53    50    F2 012E   426         AOBLSS  R0,R3,80$               ; For each page of P1 page tables
                                0132   427
    0000000C'EF    6543   DE 0132   428         MOVAL   (R5)[R3], A_SPTEND      ; SAVE END OF TABLE ADDRESS.
          50   53    01   C3 013A   429         SUBL3   #1,R3,R0                ; Decrease for base register
             0D   50    DA 013E   430         MTPR    R0, #PR$_SLR            ; SET SYSTEM LENGTH REGISTER
                                0141   431
          53   7F 8F   93 0141   432 90$:      BITB    #^X7F,R3                ; End of page?
                   0B   13 0145   433         BEQL    100$                    ; Branch if so
                 6543   D4 0147   434         CLRL    (R5)[R3]                ; Invalid PTE and no access
    EF 53    000FFFFF 8F   F3 014A   435         AOBLEQ  #^XFFFFF,R3,90$         ; Increment and branch
                                0152   436
             55    6543   DE 0152   437 100$:    MOVAL   (R5)[R3],R5             ; First free
          51   54    09   78 0156   438         ASHL    #9,R4,R1                ; P1 address
00000008'EF   51    40000400 8F   C9 015A   439         BISL3   #^X40000400,R1,A_P1BUFVA; Virtural address of P1 scratch area
                                0166   440
             51    54   D0 0166   441         MOVL    R4,R1                   ; P1 PFN to map for scratch area
       52    01800000 8F   D0 0169   442         MOVL    #PTE$M_OWN ! PTE$C_NA,R2; No access to these pages
             50    7E 8F   9A 0170   443         MOVZBL  #K_P1PAGES,R0           ; Map K_P1PAGES pages
                                0174   444
                   39   10 0174   445         BSBB    MAPMEM$IOSPACE          ; setup that page and the other 127
       51    00000008'EF   D0 0176   446         MOVL    A_P1BUFVA,R1            ; Get P1 virtural address
             00000685'EF   16 017D   447         Jsb     RPTEADR                 ; Get address of PTE                    [27]
    00000004'EF    51   D0 0183   448         MOVL    R1,A_P1BUFPTE           ; Address of first P1 PTE
             00000000'EF   16 018A   449         Jsb     MAP$IOSPACE             ; Map I/O space                         [27]
    00000000'EF    55   D0 0190   450         MOVL    R5,DS$GA_LASTADR        ; Record last available
                                0197   451
          00000000'EF   16 0197   452         JSB     MEMPOOL$INIT            ; Initialize memory pool
                                019D   453
```

```
                00000242'EF   16  019D   454          Jsb     STACKPROT                       ; Set protections on stack pages        [27]
      000002AA'EF    00       FB  01A3   455          CALLS   #0,MAPFREE                      ; Free memory
                              01AA   456
                      007C 8F  BA  01AA   457          POPR    #^M<R2,R3,R4,R5,R6>             ; Restore the saved registers           [26]
                              05  01AE   458          RSB
```

```
01AF    460          .SBTTL  MAPMEM$IOSPACE  Setup page tables for I/O space
01AF    461 ;++
01AF    462 ; FUNCTIONAL DESCRIPTION:
01AF    463 ;
01AF    464 ;       This routine is called by MAP$IOSPACE to setup a PTE
01AF    465 ;       for an I/O space address. If necessary a page is allocated for the
01AF    466 ;       corresponding SPTE and filled with invalid no access PTE's.
01AF    467 ;       The particular address desired is then given the specified
01AF    468 ;       protection.
01AF    469 ;
01AF    470 ; CALLING SEQUENCE:
01AF    471 ;
01AF    472 ;       BSBW    MAPMEM$IOSPACE
01AF    473 ;
C1AF    474 ; INPUT PARAMETERS:     NONE
01AF    475 ;
01AF    476 ; IMPLICIT INPUTS:
01AF    477 ;
01AF    478 ;       R0      Count of pages to setup
01AF    479 ;       R1      P1 PFN to map
01AF    480 ;       R2      PTE format protection and valid bits (pte format)
01AF    481 ;       R5      Address of free space pointer
01AF    482 ;
01AF    483 ; OUTPUT PARAMETERS:    NONE
01AF    484 ;
01AF    485 ; IMPLICIT OUTPUTS:
01AF    486 ;
01AF    487 ;       R5      Possibly update free page pointer
01AF    488 ;--
```

```
                                01AF    490 MAPMEM$IOSPACE::
                          18    BB  01AF    491         PUSHR   #^M<R3,R4>                      ; Save
                       53   0A    DB  01B1    492         MFPR    #PR$_P1BR,R3                    ; P1 base register
                    53    6341    DE  01B4    493         MOVAL   (R3)[R1],R3                     ; VA of P1 PTE
              54    53    15 09   EF  01B8    494         EXTZV   #9,#21,R3,R4                    ; PFN of SVA
                       7E   0C    DB  01BD    495         MFPR    #PR$_SBR,-(SP)                  ; Get system base register
                    54    9E44    DE  01C0    496         MOVAL   @(SP)+[R4],R4                   ; Physical address of SPTE
                  23 64    1F    E0  01C4    497         BBS     #31,(R4),20$                    ; Branch if already valid
                                01C8    498
                          03    BB  01C8    499         PUSHR   #^M<R0,R1>                      ; Save 2
           50     55    F7 8F   78  01CA    500         ASHL    #-9,R5,R0                       ; PFN of free page
           64   F1800000 E0    9E  01CF    501         MOVAB   PTE$M_VALID ! PTE$C_URKW ! PTE$M_OWN(R0),(R4)
                    51   7F 8F   8A  01D6    502         BICB    #^X7F,R1                        ; Lowest PFN this page of PTE's
                    50   7F A1   9E  01DA    503         MOVAB   127(R1),R0                      ; Final PFN
           85   01800000 E1    9E  01DE    504 10$:    MOVAB   PTE$M_OWN ! PTE$C_NA(R1), -
                                01E5    505                 (R5)+                           ; Fill invalid no access PTE's
                    F5 51   50  F3  01E5    506         AOBLEQ  R0,R1,10$                       ; Count and continue
                          03    BA  01E9    507         POPR    #^M<R0,R1>                      ; Restore 2
                                01EB    508
        7E  64    15    00    EF  01EB    509 20$:    EXTZV   #0,#21,(R4),-(SP)               ; Get Physical PFN of P1 PTE
        53  17    09    8E    F0  01F0    510         INSV    (SP)+,#9,#23,R3                 ; Now have Physical addr of PTE
         63   F8000000 8F    CA  01F5    511         BICL    #PTE$M_PROT!PTE$M_VALID,(R3) ; Clear protection
                 63   52    C8  01FC    512         BISL    R2,(R3)                         ; Set protection
                                01FF    513
                          18    BA  01FF    514         POPR    #^M<R3,R4>                      ; Restore
                          51    D6  0201    515         INCL    R1                              ; Update PFN
                       A9 50    F5  0203    516         SOBGTR  R0,MAPMEM$IOSPACE               ; Repeat until done all pages
                                05  0206    517         RSB
```

```
                                0207   519          .SBTTL  ACCESSABLE      check a page for accessability
                                0207   520 ;++
                                0207   521 ; FUNCTIONAL DESCRIPTION:
                                0207   522 ;
                                0207   523 ;       This routine is used to determine if every word on a page
                                0207   524 ;       can be read and written. specificially by writing each
                                0207   525 ;       QUADWORD to itself.
                                0207   526 ;
                                0207   527 ; CALLING SEQUENCE:
                                0207   528 ;
                                0207   529 ;       ACCESSABLE(address)
                                0207   530 ;
                                0207   531 ; INPUT PARAMETERS:     NONE
                                0207   532 ;
                                0207   533 ;       address Page address to check bits 0-8 are ignored
                                0207   534 ;
                                0207   535 ; IMPLICIT INPUTS:      NONE
                                0207   536 ;
                                0207   537 ; OUTPUT PARAMETERS:    NONE
                                0207   538 ;
                                0207   539 ; IMPLICIT OUTPUTS:     NONE
                                0207   540 ;
                                0207   541 ; COMPLETION CODES:
                                0207   542 ;
                                0207   543 ;       SS$_NORMAL      if page ok
                                0207   544 ;       SS$_MCHK        if machine check occured during processing
                                0207   545 ;
                                0207   546 ;--
                                0207   547
                                0207   548 ACCESSABLE:
                          0000  0207   549          .WORD   ^M<>                    ; Save no registers
                   6D   1D'AF   9E  0209   550          MOVAB   B^30$,(FP)              ; Set condition handler
51    04 AC   000001FF  8F   CB  020D   551          BICL3   #^X1FF,4(AP),R1         ; Address to start
                                0216   552
                   61   61   7D  0216   553          MOVQ    (R1),(R1)               ; Copy quadword to self         [28]
                   50   01   D0  0219   554          MOVL    S^#SS$_NORMAL,R0        ; Success
                             04  021C   555          RET
                                021D   556
                          0000  021D   557 30$:     .WORD   ^M<>                    ; Save no registers
                   50   04 AC   7D  021F   558          MOVQ    4(AP),R0                ; Signal and mechanism pointers
      04 A0    00660088  8F   D1  0223   559          CMPL    #DS$_MCHK,4(R0)         ; Machine check?
                         0F   12  0228   560          BNEQ    40$                     ; Branch if not
         0C A1    04 A0   D0  022D   561          MOVL    4(R0),12(R1)            ; Change routine R0
                         7E   7C  0232   562          CLRQ    -(SP)                   ; No args to unwind
      00000000'EF    02   FB  0234   563          CALLS   #2,SYS$UNWIND           ; Unwind stack
                             04  023B   564          RET                             ; Return
                                023C   565
            50   0918  8F   3C  023C   566 40$:     MOVZWL  #SS$_RESIGNAL,R0        ; Let someone else have it
                             04  0241   567          RET                             ; Return
```

```
                        0242   569          .SBTTL  STACKPROT
                        0242   570 ;++
                        0242   571 ; FUNCTIONAL DESCRIPTION:
                        0242   572 ;
                        0242   573 ;       This routine is used to set the protections on stack pages so
                        0242   574 ;       that underflows of each stack will cause access violations.
                        0242   575 ;       The user stack underflows into the SUPER stack, the SUPER stack
                        0242   576 ;       underflows into the EXEC stack, the EXEC stack underflows into
                        0242   577 ;       the INTERRUT stack (protected KW), the KERNEL stack underflows into
                        0242   578 ;       SCB_UNKINT which is read only.
                        0242   579 ;
                        0242   580 ; CALLING SEQUENCE:
                        0242   581 ;
                        0242   582 ;       BSBW    STACKPROT
                        0242   583 ;
                        0242   584 ; INPUT PARAMETERS:     NONE
                        0242   585 ;
                        0242   586 ; IMPLICIT INPUTS:
                        0242   587 ;
                        0242   588 ;       The page tables and stack addresses
                        0242   589 ;
                        0242   590 ; OUTPUT PARAMETERS:    NONE
                        0242   591 ;
                        0242   592 ; IMPLICIT OUTPUTS:     NONE
                        0242   593 ;
                        0242   594 ; COMPLETION CODES:     NONE
                        0242   595 ;
                        0242   596 ; SIDE EFFECTS:         NONE
                        0242   597 ;--
                        0242   598
                        0242   599 STACKPROT:
     51   00000000'EF   DE   0242   600          MOVAL   P0PT_BASE,R1            ; Base P0 page table
          50    0000'8F   3C   0249   601          MOVZWL  #STAC' _BASE@-9,R0      ; PFN of read only section
                          024E   602
   6140    78000000 8F   CA   024E   603 10$:     BICL2   #PTE$M_PROT,(R1)[R0]    ; Clear protections in all pte's
 F0 50    00000000'8F   F2   0256   604          AOBLSS  #USTKPTR@-9,R0,10$      ; Branch if more to clear
                          025E   605
          50    0000'8F   3C   025E   606          MOVZWL  #STACK_BASE@-9,R0       ; PFN of read only section
                          0263   607
        6140    00        C8   0263   608          BISL    #PTE$C_NA, (R1)[R0]     ; Set protection to no access
                50        D6   0267   609          INCL    R0                      ; 1 page protects bottom of stack
                          0269   610
   6140    70000000 8F   C8   0269   611 30$:     BISL    #PTE$C_URKW, (R1)[R0]   ; Set protection to KERNEL Write
 F0 50    00000000'8F   F2   0271   612          AOBLSS  #ISTKPTR@-9,R0,30$      ; Protect KW thru INTERRUPT stack
                          0279   613
   6140    68000000 8F   C8   0279   614 40$:     BISL    #PTE$C_UREW, (R1)[R0]   ; Set protection to EXEC write
 F0 50    00000000'8F   F2   0281   615          AOBLSS  #ESTKPTR@-9,R0,40$      ; Protect EW thru EXEC stack
                          0289   616
   6140    60000000 8F   C8   0289   617 50$:     BISL    #PTE$C_URSW, (R1)[R0]   ; Set protection to SUPER write
 F0 50    00000000'8F   F2   0291   618          AOBLSS  #SSTKPTR@-9,R0,50$      ; Protect SW thru SUPER stack
                          0299   619
   6140    20000000 8F   C8   0299   620 60$:     BISL    #PTE$C_UW, (R1)[R0]     ; Set protection to USER write
 F0 50    00000000'8F   F2   02A1   621          AOBLSS  #USTKPTR@-9,R0,60$      ; Protect UW thru USER stack
                          02A9   622
                          05   02A9   623          RSB                             ; Return
```

```
                        .SBTTL  MAP FREE MEMORY PROCEDURE.
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       MAPS ALL FREE MEMORY BEFORE STARTING A PROGRAM.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       PROCEDURE CALL.
                ;
                ; INPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       L$L_HPTL        - LENGTH OF EACH HARDWARE PARAMETER TABLE.
                ;       L$A_LASTADR     - LAST ADDRESS OF USER PROGRAM.
                ;       DSA$GL_UNITS    - NUMBER OF UNITS CURRENTLY SELECTED.
                ;       L_POLEN         - LENGTH OF P0 PAGE TABLE.
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;
                ;       EFFECTIVE $DS_RELBUF_L OF ALL USER BUFFER SPACE.
                ;
                ;--
```

```
                     003C  02AA   664 .ENTRY  MAPFREE,^M<R2,R3,R4,R5>          ; ENTRY POINT
                           02AC   665
          54  00000000'EF   9E   02AC   666          MOVAB   DS$K_PRGSIZ+DS$A_PRGBGN,R4 ; Assume no program loaded
              00000000'EF   16   02B3   667          JSB     DSR$CheckLoad          ; Find out, check program hdr      [25]
                    11 50   E9   02B9   668          BLBC    R0, 5$                 ; Branch if no program loaded      [25]
                           02BC   669
                           02BC   670          IFNORD  #4,L$A_LASTADR,5$      ; Branch if can't read it
          54  00000214'EF   D0   02C6   671          MOVL    L$A_LASTADR, R4        ; Use size of program instead
                           02CD   672 5$:
                      53   D4   02CD   673          CLRL    R3                     ; CLEAR R3.
                 52   53   09   78   02CF   674 10$:     ASHL    #9,R3,R2               ; MAKE VIRTURAL ADDRESS THIS PAGE
                      54   52   D1   02D3   675          CMPL    R2, R4                 ; CHECK FOR END OF USER CODE.
                           20   19   02D6   676          BLSS    30$
          0000FA00 8F      52   D1   02D8   677          CMPL    R2, #DSA$AT_APTTXT     ; Bottom of APT text area?
                           09   19   C2DF   678          BLSS    20$
          00000000'EF      52   D1   02E1   679          CMPL    R2, L^DS$GA_LASTADR    ; Check for start of buffers
                           0E   19   02E8   680          BLSS    30$
00000000'EF43     81800000 8F   CA   02EA   681 20$:     BICL2   #PTE$M_VALID + PTE$M_OWN, -    ; INVALIDATE PAGE.
                                02F6   682                    POPT_BASE[R3]
                           0C   11   02F6   683          BRB     40$
00000000'EF43     81800000 8F   C8   02F8   684 30$:     BISL2   #PTE$M_VALID + PTE$M_OWN, -    ; VALIDATE PAGE.
                                0304   685                    POPT_BASE[R3]
                                0304   686
        C5 53     00000000'EF   F2   0304   687 40$:     AOBLSS  L^L_POLEN, R3, 10$             ; CHECK FOR ALL DONE.      [27]
                                030C   688
                                030C   689 ;
                                030C   690 ; INIT P1 SPACE BUFFER.
                                030C   691 ;
                                030C   692
             52   7E 8F   9A   030C   693          MOVZBL  #K_P1PAGES,R2          ; P1 PTE's to clear
          53  00000004'EF   D0   0310   694          MOVL    A_P1BUFPTE, R3                 ; PAGE TABLE ENTRY POINTER.
                           0317   695
                      83   D4   0317   696 50$:     CLRL    (R3)+                          ; FREE PAGE.
                    FB 52  F5   0319   697          SOBGTR  R2,50$
                           031C   698
                           031C   699 ;
                           031C   700 ; INIT SYSTEM SPACE BUFFER.
                           031C   701 ;
                           031C   702
          53  00000000'EF   D0   031C   703          MOVL    MMG$GL_SPTBASE,R3              ; System page table pointer.
                           0323   704
       03 63  02      17   ED   0323   705 60$:     CMPZV   #PTE$V_OWN, #PTE$S_OWN, (R3), #3  ; SUPERVISOR PAGE?
                           02   13   0328   706          BEQL    70$
                           63   D4   032A   707          CLRL    (R3)                   ; FREE PAGE.
                                032C   708
FFED 53    04     0000000C'EF   F1   032C   709 70$:     ACBL    A_SPTEND, #4, R3, 60$          ; LOOP TO END.
           50  00000000'EF      7E   0336   710          MOVAQ   DS$GL_BUFCNT,R0        ; Address of buffer counts
                           80   7C   033D   711          CLRQ    (R0)+                  ; Reset P0, P1 buffer counts
                           80   7C   033F   712          CLRQ    (R0)+                  ; RESET SYS AND SUP BUFFER CNTS.
                                0341   713
                    FEFD CF   16   0341   714          Jsb     STACKPROT              ; Re-protect stacks                [27]
                           04   0345   715          RET
```

```
                        717          .SBTTL  GET A BLOCK OF VIRTUAL MEMORY.
                        718 ;++
                        719 ; FUNCTIONAL DESCRIPTION:
                        720 ;
                        721 ;       THIS PROCEDURE EMULATES THE STARLET SYSTEM SERVICE "EXPREG",
                        722 ;       EXPAND PROGRAM REGION.
                        723 ;
                        724 ; CALLING SEQUENCE:
                        725 ;
                        726 ;       STARLET PROCEDURE CALL.
                        727 ;
                        728 ; INPUT PARAMETERS:
                        729 ;
                        730 ;       EXPREG$_PAGCNT(AP) =    NUMBER OF PAGES TO ADD TO THE END OF THE PROGRAM
                        731 ;                               REGION.
                        732 ;       EXPREG$_RETADR(AP) =    ADDRESS OF A 2-LONGWORD ARRAY IN WHICH TO RETURN
                        733 ;                               THE VIRTUAL ADDRESSES OF THE STARTING PAGE AND
                        734 ;                               ENDING PAGE OF THE EXPANTION AREA.
                        735 ;       EXPREG$_ACMODE =        ACCESS MODE.  NOT USED.
                        736 ;       EXPREG$_REGION(AP) =    REGION CODE:
                        737 ;                                       0 = PROGRAM (P0) REGION.
                        738 ;                                       1 = CONTROL (P1) REGION.
                        739 ;                                       2 = SYSTEM REGION.
                        740 ;
                        741 ; IMPLICIT INPUTS:
                        742 ;
                        743 ;       NONE
                        744 ;
                        745 ; OUTPUT PARAMETERS:
                        746 ;
                        747 ;       @EXPREG$_RETADR(AP) =   2-LONGWORD ARRAY CONTAINING THE STARTING AND
                        748 ;                               ENDING ADDRESSES OF THE EXPANDED REGION.
                        749 ;
                        750 ; IMPLICIT OUTPUTS:
                        751 ;
                        752 ;       NONE
                        753 ;
                        754 ; COMPLETION CODES:
                        755 ;
                        756 ;       SS$_NORMAL =    SERVICE SUCESSFULLY COMPLETED.
                        757 ;       SS$_ILLPAGCNT = PAGE COUNT < 1.
                        758 ;       SS$_VASFULL =   VIRTUAL ADDRESS SPACE FULL.
                        759 ;
                        760 ; SIDE EFFECTS:
                        761 ;
                        762 ;       NONE
                        763 ;
                        764 ; REGISTER USAGE:
                        765 ;
                        766 ;       R0 =    RETURN STATUS.
                        767 ;       R1 =    BEGINNING P0 PTE POINTER.
                        768 ;       R2 =    P0 PTE POINTER LIMIT.
                        769 ;       R3 =    COUNT / 'REGION' PTE POINTER.
                        770 ;       R4 =    COUNT / 'REGION' PTE LIMIT.
                        771 ;       R5 =    VIRTUAL ADDRESS.
                        772 ;       R6 =    PTE DATA.
                        773 ;--
```

```
                        007C  0346   775 .Entry  Crd$ExpReg,^M<R2,R3,R4,R5,R6>          ; CRD entry point                [22]
                              0348   776
    52  00000000'EF   F7 8F   78  0348   777          AshL    #-9, L^DS$GA_LastAdr, R2    ; Calculate PFN of LastAdr       [22]
    54  00000000'EF    52     C3  0351   778          SubL3   R2, L^L_POLen, R4           ; Get remaining length of table  [22]
    52  00000000'EF42         DE  0359   779          MovAL   POPT_Base [R2], R2          ; Get new starting address       [22]
                  10          11  0361   780          Brb     ExpReg_Common               ; Skip to the common stuff       [22]
                              0363   781
                        007C  0363   782 .ENTRY  EXE$EXPREG,^M<R2,R3,R4,R5,R6>   ; ENTRY POINT
                              0365   783
    52  00000000'EF           DE  0365   784          MOVAL   POPT_BASE, R2               ; START OF PO PAGE TABLES.
    54  00000000'EF           D0  036C   785          MOVL    L^L_POLEN, R4               ; PO PAGE TABLE LENGTH.          [27]
                              0373   786
                              0373   787 ExpReg_Common:                                 ; Common point                   [22]
                              0373   788
          51    52            D0  C373   789 10$:     MOVL    R2, R1                      ; SAVE PAGE NUMBER IN CASE FREE.
      53  04 AC              D0  0376   790          MOVL    EXPREG$_PAGCNT(AP), R3      ; GET PAGE LIMIT.
            08                14  037A   791          BGTR    20$                         ; BR IF > 1.
  50  00FC 8F                3C  037C   792          MOVZWL  #SS$_ILLPAGCNT, R0          ; ILLEGAL PAGE COUNT.
        00E2                 31  0381   793          BRW     EXE$EXPREG_X
        07 54                F5  0384   794 20$:     SOBGTR  R4, 30$                     ; COUNT PAGES.
  50  0244 8F                3C  0387   795          MOVZWL  #SS$_VASFULL, R0            ; OUT OF MEMORY.
            0A                11  038C   796          BRB     40$
            82                D5  038E   797 30$:     TSTL    (R2)+                       ; LOOK FOR EXISTS BUT NOT VALID.
            F1                13  0390   798          BLEQ    10$                         ; BR ON NO GOOD.
          EF  53              F5  0392   799          SOBGTR  R3, 20$                     ; LOOP TILL ENOUGH PAGES.
        50    01              D0  0395   800          MOVL    #SS$_NORMAL, R0             ; NORMAL RETURN.
      55    61    09          78  0398   801 40$:     ASHL    #9, (R1), R5                ; GET PO VIRTUAL ADDRESS.
00000000'EF     55            D0  039C   802          MOVL    R5, DS$GQ_PHYADR            ; SAVE THE PHYSICAL ADR.
                7E            7C  03A3   803          CLRQ    -(SP)                       ; SPACE FOR VIRTUAL ADRS.
 03   00   10 AC              CF  03A5   804          CASEL   EXPREG$_REGION(AP), #0, #3  ; SET UP FOR REGION.
                        000D'       03AA   805 50$:     .WORD   60$ - 50$                   ; P0
                        0012'       03AC   806          .WORD   70$ - 50$                   ; P1
                        0027'       03AE   807          .WORD   80$ - 50$                   ; SYS
                        000D'       03B0   808          .WORD   60$ - 50$                   ; SUPERVISOR.
                              03B2   809
              50          D4  03B2   810          CLRL    R0                          ; RETURN ERROR STATUS.
          00AF            31  03B4   811          BRW     EXE$EXPREG_X
                              03B7   812
        53    51            7D  03B7   813 60$:     MOVQ    R1, R3                      ; COPY PO PAGE TABLE POINTERS.
              3D            11  03BA   814          BRB     110$                        ; BR TO COMMON CODE.
                              03BC   815
53  00000004'EF             D0  03BC   816 70$:     MOVL    A_P1BUFPTE, R3              ; P1 PAGE TABLE POINTER.
    54  01F8 C3             DE  03C3   817          MOVAL   K_P1PAGES*4(R3), R4         ; End of P1 page table
55  00000008'EF             D0  03C8   818          MOVL    A_P1BUFVA, R5               ; P1 BUFFER VIRTUAL ADDRESS.
              12            11  03CF   819          BRB     100$                        ; BR TO COMMON CODE.
                              03D1   820
53  00000000'EF             D0  03D1   821 80$:     MOVL    MMG$GL_SPTBASE,R3           ; System page table pointer
54  0000000C'EF             D0  03D8   822          MOVL    A_SPTEND, R4                ; END OF SYS PAGE TABLE.
    55  01    1F            78  03DF   823          ASHL    #31, #1, R5                 ; SYSTEM VIRTUAL ADDRESS.
```

```
                                63   D5  03E3    825 100$:   TSTL    (R3)                            ; LOOK FOR VALID BIT.
                                12   18  03E5    826         BGEQ    110$                            ; BR IF NOT VALID.
                 55    00000200 8F   C0  03E7    827         ADDL    #512, R5                        ; NEXT VIRTUAL PAGE.
               FFEF 53     04    54   F1  03EE    828         ACBL    R4, #4, R3, 100$                ; LOOP TO END OF PT.
                           6E    55   D0  03F4    829         MOVL    R5, (SP)                        ; SAVE FIRST ADR.
                                36   11  03F7    830         BRB     130$
                                         03F9    831
                           6E    55   D0  03F9    832 110$:   MOVL    R5, (SP)                        ; SAVE FIRST ADR.
                        56 01    1F   78  03FC    833         ASHL    #PTE$V_VALID, #1, R6            ; GET VALID BIT.
                56 02    17    10 AC   F0  0400    834         INSV    EXPREG$_REGION(AP), -           ; INSERT REGION CODE
                                         0406    835                   #PTE$V_OWN, #PTE$S_OWN, R6    ;  INTO OWN FIELD.
                56    20000000 8F   C8  0406    836         BISL2   #PTE$C_UW, R6                   ; User writeable
                                         040D    837
                61    78000000 8F   CA  040D    838 120$:   BICL2   #PTE$M_PROT, (R1)               ; Clear old protection
                           61   56   C8  0414    839         BISL2   R6, (R1)                        ; Set valid, prot, and region code
                           81   81   D0  0417    840         MOVL    (R1)+, (R3)+                    ; COPY TO 'REGION' PTE.
                55    00000200 8F   C0  041A    841         ADDL    #512, R5                        ; NEXT PHYSICAL PAGE.
                           52   51   D1  0421    842         CMPL    R1, R2                          ; CHECK FOR END OF BUFFER.
                                0E   18  0424    843         BGEQ    140$
                           54   53   D1  0426    844         CMPL    R3, R4                          ; CHECK FOR END OF REGION.
                                04   18  0429    845         BGEQ    130$                            ; ERROR EXIT.
                                63   D5  042B    846         TSTL    (R3)                            ; CHECK NEXT PAGE FOR FREE.
                                DE   18  042D    847         BGEQ    120$                            ; LOOP TIL ENOUGH.
                                         042F    848
                     50   0244 8F   3C  042F    849 130$:   MOVZWL  #SS$_VASFULL, R0                ; VIRTUAL ADDRESS SPACE FULL.
                                         0434    850
                  04 AE   55   01   C3  0434    851 140$:   SUBL3   #1, R5, 4(SP)                   ; STORE END ADDRESS.
          00000004'EF     71   09   78  0439    852         ASHL    #9, -(R1), DS$GQ_PHYADR+4       ; SAVE END PHYSICAL ADR.
          00000004'EF   01FF 8F   A8  0441    853         BISW2   #^X1FF, DS$GQ_PHYADR+4          ; SET TO LAST BYTE ON PAGE
                     08 AC   D5   044A    854         TSTL    EXPREG$_RETADR(AP)              ; CHECK IF RETURN ADR REQUESED.
                           17   13  044D    855         BEQL    EXE$EXPREG_X                    ; SKIP IF NOT.
                  04 AE 01FF 8F   A8  044F    856         BISW2   #^X1FF, 4(SP)                   ; SET TO HIGHEST BYTE ADDR
                     10 AC   01   D1  0455    857         CMPL    #1,EXPREG$_REGION(AP)           ; P1 space?
                           07   12  0459    858         BNEQ    150$                            ; Branch if not
                        51 8E   7D  045B    859         MOVQ    (SP)+,R1                        ; Get low, high
                           51   DD  045E    860         PUSHL   R1                              ; put low in first
                           52   DD  0460    861         PUSHL   R2                              ; put high in second
                                         0462    862
                     08 BC   8E   7D  0462    863 150$:   MOVQ    (SP)+, @EXPREG$_RETADR(AP)      ; STORE VIRTUAL ADDRESSES.
                                         0466    864 EXE$EXPREG_X:
                                04   0466    865         RET                                     ; RETURN.
```

```
0467   867          .SBTTL  RELEASE A BLOCK OF VIRTUAL MEMORY.
0467   868 ;++
0467   869 ; FUNCTIONAL DESCRIPTION:
0467   870 ;
0467   871 ;       THIS PROCEDURE EMULATES THE STARLET SYSTEM SERVICE ''CNTREG'',
0467   872 ;       CONTRACT PROGRAM REGION.
0467   873 ;
0467   874 ; CALLING SEQUENCE:
0467   875 ;
0467   876 ;       STARLET PROCEDURE CALL.
0467   877 ;
0467   878 ; INPUT PARAMETERS:
0467   879 ;
0467   880 ;       CNTREG$_PAGCNT(AP) =    NUMBER OF PAGES TO REMOVE FROM THE END OF THE
0467   881 ;                               PROGRAM REGION.
0467   882 ;       CNTREG$_RETADR(AP) =    ADDRESS OF A 2-LONGWORD ARRAY IN WHICH  TO RETURN
0467   883 ;                               THE VIRTUAL ADDRESSES OF THE STARTING PAGE AND
0467   884 ;                               ENDING PAGE OF THE REMOVED AREA.
0467   885 ;       CNTREG$_ACMODE(AP) =    ACCESS MODE. IGNORED.
0467   886 ;       CNTREG$_REGION(AP) =    REGION CODE:
0467   887 ;                                       0 = PROGRAM (P0) REGION.
0467   888 ;                                       1 = CONTROL (P1) REGION.
0467   889 ;                                       2 = SYSTEM REGION.
0467   890 ;
0467   891 ; IMPLICIT INPUTS:
0467   892 ;
0467   893 ;       NONE
0467   894 ;
0467   895 ; OUTPUT PARAMETERS:
0467   896 ;
0467   897 ;       @CNTREG$_RETADR(AP) =   2-LONGWORD ARRAY CONTAINING THE STARTING AND
0467   898 ;                               ENDING ADDRESSES OF THE REMOVEDED REGION.
0467   899 ;
0467   900 ; IMPLICIT OUTPUTS:
0467   901 ;
0467   902 ;       NONE
0467   903 ;
0467   904 ; COMPLETION CODES:
0467   905 ;
0467   906 ;       SS$_NORMAL =    SERVICE SUCESSFULLY COMPLETED.
0467   907 ;       SS$_ILLPAGCNT = PAGE COUNT < 1.
0467   908 ;       SS$_PAGOWNVIO = PAGE OWNED BY MORE PRIVILAGED ACCESS MODE.
0467   909 ;
0467   910 ; SIDE EFFECTS:
0467   911 ;
0467   912 ;       NONE
0467   913 ;
0467   914 ; REGISTER USAGE:
0467   915 ;
0467   916 ;       R0 =    RETURN STATUS CODE.
0467   917 ;       R1 =    'REGION' END PTE.
0467   918 ;       R2 =    'REGION' BEGINNING PTE.
0467   919 ;       R3 =    VIRTUAL ADDRESS.
0467   920 ;       R4 =    PAGE COUNTER.
0467   921 ;       R5 =    POPT POINTER.
0467   922 ;--
```

```
                                003C  0467    924          .ENTRY  EXE$CNTREG,^M<R2,R3,R4,R5>      ; ENTRY POINT
                                      0469    925
                          50    01    D0    0469    926          MOVL    #SS$_NORMAL, R0                 ; START ASSUMING THE BEST!
                    03    00    10 AC    CF    046C    927          CASEL   CNTREG$_REGION(AP), #0, #3      ; DISPATCH ON REGION CODE.
                                      0471    928
                                000C'       0471    929 10$:     .WORD   20$ - 10$                       ; P0
                                0024'       0473    930          .WORD   30$ - 10$                       ; P1
                                003E'       0475    931          .WORD   40$ - 10$                       ; SYS
                                000C'       0477    932          .WORD   20$ - 10$                       ; SUPERVISOR.
                                      0479    933
                             18 10 AC    E8    0479    934          BLBS    CNTREG$_REGION(AP), 30$         ; REGION INDICATOR.
                                      047D    935
                   51    00000000'EF    D0    047D    936 20$:     MOVL    L^P0LEN, R1                     ; P0 PAGE COUNT.
                         53    51    09    78    0484    937          ASHL    #9, R1, R3                      ; END VIRTUAL ADDRESS.
                   52    00000000'EF    DE    C488    938          MOVAL   P0PT_BASE, R2                   ; POINTER TO TOP OF PAGE TABLE.
                           51    6241    DE    048F    939          MOVAL   (R2)[R1], R1                    ; POINTER TO END OF PAGE TABLE.
                                 35    11    0493    940          BRB     50$                             ; BR TO COMMON CODE.
                                      0495    941
                   52    00000004'EF    D0    0495    942 30$:     MOVL    A_P1BUFPTE, R2                  ; START OF P1 PTE.
                         51    01F8 C2    DE    049C    943          MOVAL   K_P1PAGES*4(R2), R1             ; End of P1 page table
53    00000008'EF    0000FC00 8F    C1    04A1    944          ADDL3   #K_P1PAGES@9,A_P1BUFVA,R3       ; LAST VIRTURAL ADDRESS
                                 1B    11    04AD    945          BRB     50$                             ; BR TO COMMON CODE.
                                      04AF    946
                   51    0000000C'EF    D0    04AF    947 40$:     MOVL    L^A_SPTEND, R1                  ; END OF SYS PAGE TABLE.        [27]
                   52    00000000'EF    D0    04B6    948          MOVL    MMG$GL_SPTBASE, R2              ; Start of SYS page table
                         53    01    1F    78    04BD    949          ASHL    #31, #1, R3                     ; START OF SYS VIRTUAL ADR.
                         7E    51    52    C3    04C1    950          SUBL3   R2, R1, -(SP)                   ; SPT LENGTH.
                   53    16    07    8E    F0    04C5    951          INSV    (SP)+, #9-2, #20+2, R3          ; COMPUTE LAST VIRTUAL ADDRESS.
```

```
                        53   D7  04CA   953 50$:    DECL    R3                                      ; END OF PREVIOUS PAGE.
                        54   D4  04CC   954          CLRL    R4                                      ; INIT PAGE COUNTER.
                                 04CE   955
                        71   D5  04CE   956 60$:    TSTL    -(R1)                                   ; LOOK FOR VALID BIT.
                        53   18  04D0   957          BGEQ    100$                                    ; BR IF NOT VALID.
                                 04D2   958
      10 AC   61   02   17   ED  04D2   959          CMPZV   #PTE$V_OWN, #PTE$S_OWN, (R1), -         ; CHECK 'OWN' CODE
                                 04D8   960                    CNTREG$_REGION(AP)                    ;  FOR REGION.
                        4B   12  04D8   961          BNEQ    100$                                    ; WRONG OWNER.
                                 04DA   962
   55   61    FFF00000 8F   CB  04DA   963          BICL3   #^CPTE$M_PFN!^X100000, (R1), R5         ; ISOLATE PFN
      55    00000000'EF45   DE  04E2   964          MOVAL   POPT_BASE[R5], R5                       ; GET PO PTE, SAME PAGE.
                  55   51   D1  04EA   965          CMPL    R1, R5                                  ; CHECK FOR PO REGION.
                        22   13  04ED   966          BEQL    80$                                     ; BR IF PO.
            7E    65   61   CD  C4EF   967          XORL3   (R1),(R5), -(SP)                        ; GET DIFFERENCE BETWEEN PTE'S
         8E    827FFFFF 8F   D3  04F3   968          BITL    #^C<PTE$M_MODIFY ! -                    ; Ignore MODIFY,                [19]
                                 04FA   969                      PTE$M_PROT !   -                  ; PROT,                         [19]
                                 04FA   970                      PTE$M_OWN>,    -                  ;  and OWN fields               [19]
                                 04FA   971                      (SP)+                             ; Do any useful fields differ?
                        13   13  04FA   972          BEQL    70$                                     ; NO, RELEASE IT
                                 04FC   973          ERRSUP_S                                        ; BUM PAGE TABLES.
                        2D   11  050D   974          BRB     120$
                                 050F   975
                        61   D4  050F   976 70$:    CLRL    (R1)                                    ; RELEASE 'REGION' PAGE.
                                 0511   977
         65   81800000 8F   CA  0511   978 80$:    BICL2   #PTE$M_VALID ! PTE$M_OWN, (R5)          ; RELEASE PO PAGE.
                        54   D5  0518   979          TSTL    R4                                      ; FIRST PAGE?
                        02   12  051A   980          BNEQ    90$                                     ; BR OF NOT.
                        53   DD  051C   981          PUSHL   R3                                      ; SAVE VIRTUAL ADDR.
                                 051E   982
      0D 54    04   AC   F2  051E   983 90$:    AOBLSS  CNTREG$_PAGCNT(AP), R4, 110$            ; COUNT PAGES.
                        1E   11  0523   984          BRB     130$                                    ; SUCCESS EXIT.
                                 0525   985
                        54   D5  0525   986 100$:    TSTL    R4                                      ; FIRST PAGE?
                        07   13  0527   987          BEQL    110$                                    ; SKIP IF SO.
   50   00660080 8F   D0  0529   988          MOVL    #DS$_FRAGBUF, R0                                ; WARN CALLER.
                                 0530   989
   53   00000200 8F   C2  0530   990 110$:    SUBL    #512, R3                                        ; ADJUST VITRUAL ADR.
                  52   51  D1  0537   991          CMPL    R1, R2                                  ; CHECK FOR OUT OF PAGE TABLE.
                        92   1A  053A   992          BGTRU   60$                                     ; BR IF MORE.
                                 053C   993
   50   000001EC 8F   D0  053C   994 120$:    MOVL    #SS$_PAGOWNVIO, R0                              ; RAN OUT OF PAGES.
                                 0543   995
            08   AC   D5  0543   996 130$:    TSTL    CNTREG$_RETADR(AP)                              ; CHECK FOR RETURN REQUESTED.
                        0E   13  0546   997          BEQL    EXE$CNTREG_X                            ; EXIT IF NOT.
      53    01FF 8F   AA  0548   998          BICW2   #^X1FF, R3                                      ; BEGINNING OF PAGE.
                        53   DD  054D   999          PUSHL   R3                                      ; SAVE FIRST VIRTUAL ADR.
         08 BC    8E   7D  054F  1000          MOVQ    (SP)+, @CNTREG$_RETADR(AP)                      ; RETURN VIRTUAL ADDRESSES.
                                 0553  1001
                                 0553  1002 EXE$CNTREG_X:
                        04   0553  1003          RET                                                 ; RETURN.
```

```
                                0554  1005          .SBTTL  TURN MEMORY MANAGEMENT ON.
                                0554  1006 ;++
                                0554  1007 ; FUNCTIONAL DESCRIPTION:
                                0554  1008 ;
                                0554  1009 ;       This routine can be called by the program to enable memory management.
                                0554  1010 ;       All memory mapping, page tables and base and length registers
                                0554  1011 ;       must be setup prior to calling this routine.
                                0554  1012 ;
                                0554  1013 ; CALLING SEQUENCE:
                                0554  1014 ;
                                0554  1015 ;       NONE
                                0554  1016 ;       CALLS   #0,DS$MMON
                                0554  1017 ;
                                0554  1018 ; INPUT PARAMETERS:
                                0554  1019 ;
                                0554  1020 ;       NONE
                                0554  1021 ;
                                0554  1022 ; IMPLICIT INPUTS:
                                0554  1023 ;
                                0554  1024 ;       NONE
                                0554  1025 ;
                                0554  1026 ; OUTPUT PARAMETERS:
                                0554  1027 ;
                                0554  1028 ;       NONE
                                0554  1029 ;
                                0554  1030 ; IMPLICIT OUTPUTS:
                                0554  1031 ;
                                0554  1032 ;       NONE
                                0554  1033 ;
                                0554  1034 ; COMPLETION CODES:
                                0554  1035 ;
                                0554  1036 ;       SS$_WASSET      If was enabled before
                                0554  1037 ;       SS$_WASCLR      If was not enabled
                                0554  1038 ;
                                0554  1039 ; SIDE EFFECTS:
                                0554  1040 ;
                                0554  1041 ;       Memory management is enabled
                                0554  1042 ;
                                0554  1043 ;--
                                0554  1044
                          0000  0554  1045 .ENTRY  DSX$MMON,^M<>                  ; ENTRY POINT
                                0556  1046
                     50   01   D0  0556  1047          MOVL    #1,R0              ; Enable memory managment
                          21   10  0559  1048          BSBB    DSR$MMENABLE       ; Do it
                50   50   03   78  055B  1049          ASHL    #3,R0,R0           ; Position old enabled bit to bit 3
                          50   D6  055F  1050          INCL    R0                 ; Make it success return code
                               0561  1051 DSX$MMON_X:
                               04  0561  1052          RET
```

```
                                0562  1054          .SBTTL  TURN MEMORY MANAGEMENT OFF.
                                0562  1055 ;++
                                0562  1056 ; FUNCTIONAL DESCRIPTION:
                                0562  1057 ;
                                0562  1058 ;       This routine can be called to disable memory managment
                                0562  1059 ;       However, if the operator enabled memory management with the
                                0562  1060 ;       SET  MM ON command, it will not disable memory management.
                                0562  1061 ;
                                0562  1062 ; CALLING SEQUENCE:
                                0562  1063 ;
                                0562  1064 ;       CALLS   #0,DS$MMOFF
                                0562  1065 ;
                                0562  1066 ; INPUT PARAMETERS:
                                0562  1067 ;
                                0562  1068 ;       NONE
                                0562  1069 ;
                                0562  1070 ; IMPLICIT INPUTS:
                                0562  1071 ;
                                0562  1072 ;       NONE
                                0562  1073 ;
                                0562  1074 ; OUTPUT PARAMETERS:
                                0562  1075 ;
                                0562  1076 ;       NONE
                                0562  1077 ;
                                0562  1078 ; IMPLICIT OUTPUTS:
                                0562  1079 ;
                                0562  1080 ;       NONE
                                0562  1081 ;
                                0562  1082 ; COMPLETION CODES:
                                0562  1083 ;
                                0562  1084 ;       SS$_WASSET      If was enabled before
                                0562  1085 ;       SS$_WASCLR      If was not enabled
                                0562  1086 ;       DS$_WARNING     If was set by operator and you can't disable it
                                0562  1087 ;
                                0562  1088 ; SIDE EFFECTS:
                                0562  1089 ;
                                0562  1090 ;       NONE
                                0562  1091 ;       Memory management is disabled unless the operator explicitly
                                0562  1092 ;       enabled it.
                                0562  1093 ;
                                0562  1094 ;--
                          0000  0562  1095 .ENTRY  DSX$MMOFF,^M<>                 ; ENTRY POINT
                                0564  1096
 50   00660000 8F      D0      0564  1097          MOVL    #DS$_WARNING,R0        ; Prepare for can't disable return
      0000001C'EF      95      056B  1098          TSTB    DS$GB_MM_ENB           ; Was MM enabled by operator?
                       08      12    0571  1099          BNEQ    DSX$MMOFF_X            ; Yes, Do not disable it
                       07      10    0573  1100          BSBB    DSR$MMENABLE           ; and do it
     50   50         03      78    0575  1101          ASHL    #3,R0,R0               ; Position old enable bit to bit 3
                       50      D6    0579  1102          INCL    R0                     ; Make it success return code
                                     057B  1103 DSX$MMOFF_X:
                             04      057B  1104          RET
```

```
                                057C  1106          .SBTTL  DSR$MMENABLE    Enable to disable memory management
                                057C  1107 ;++
                                057C  1108 ; FUNCTION DESCRIPTION:
                                057C  1109 ;
                                057C  1110 ;       This routine is called to enable or disable memory managment.
                                057C  1111 ;
                                057C  1112 ; CALLING SEQUENCE:
                                057C  1113 ;
                                057C  1114 ;       BSBW    DSR$MMENABLE
                                057C  1115 ;
                                057C  1116 ; INPUT PARAMETERS:
                                057C  1117 ;
                                057C  1118 ;       R0      value to be written in to PR$_MAPEN
                                057C  1119 ;
                                C57C  1120 ; IMPLICIT INPUTS:      NONE
                                057C  1121 ;
                                057C  1122 ; OUTPUT PARAMETERS:
                                057C  1123 ;
                                057C  1124 ;       R0      Previous contents of PR$_MAPEN
                                057C  1125 ;
                                057C  1126 ; IMPLICIT OUTPUTS:     NONE
                                057C  1127 ;
                                057C  1128 ; COMPLETION CODES:     NONE
                                057C  1129 ;
                                057C  1130 ; SIDE EFFECTS:
                                057C  1131 ;
                                057C  1132 ;       Memory management is enabled or disabled.
                                057C  1133 ;--
                                057C  1134
                                057C  1135 DSR$MMENABLE::
                                057C  1136          CMK     MMENABLE                ; Change to Kernel mode
                          05    058B  1137          RSB                             ; Return to caller
                                058C  1138
                                058C  1139 CMK$MMENABLE::
                39  00    DA   058C  1140          MTPR    #0,#PR$_TBIA            ; Invalidate the TB before enabling MM
                7E  38    DB   058F  1141          MFPR    #PR$_MAPEN,-(SP)        ; Get current state of memory managment
50  50          01  00    EF   0592  1142          EXTZV   #0,#1,R0,R0             ; Only 1 bit
                38  5C    DA   0597  1143          MTPR    R0,#PR$_MAPEN           ; Set new desired state
                    01    BA   059A  1144          POPR    #^MR0                   ; Get old state
                          02   059C  1145          REI                             ; Return to previous mode
```

```
                        1147          .SBTTL  SET PROTECTION ON PAGES.
                        1148 ;++
                        1149 ; FUNCTIONAL DESCRIPTION:
                        1150 ; SET THE ACCESS PROTECTION ON A VIRTUAL PAGE
                        1151 ;
                        1152 ; CALLING SEQUENCE:
                        1153 ;
                        1154 ;       PROCEDURE CALL.
                        1155 ;
                        1156 ; INPUT PARAMETERS:
                        1157 ;
                        1158 ;       SETPRT$_INADR(AP) =     ARAY ADDRESS (INPUT)
                        1159 ;       SETPRT$_RETADR(AP) =    ARAY ADDRESS (OUTPUT)
                        1160 ;       SETPRT$_ACMODE(AP) =    ACCESS MODE (OPTIONAL)
                        1161 ;       SETPRT$_PROT(AP) =      NEW PROTECTION (BITS <3:0>)
                        1162 ;       SETPRT$_PRVPRT(AP) =    ADDRESS TO STORE PREVIOUS PROTECTION
                        1163 ;
                        1164 ; IMPLICIT INPUTS:
                        1165 ;
                        1166 ;       NONE
                        1167 ;
                        1168 ; OUTPUT PARAMETERS:
                        1169 ;
                        1170 ;       R0 = COMPLETION CODE
                        1171 ;
                        1172 ; IMPLICIT OUTPUTS:
                        1173 ;
                        1174 ;       NONE
                        1175 ;
                        1176 ; COMPLETION CODES:
                        1177 ;
                        1178 ;       DS$_NORMAL = SERVICE SUCCESSFULLY COMPLETED
                        1179 ;       SS$_ACCVIO = ACCESS VIOLATION
                        1180 ;       SS$_LENVIO = LENGTH VIOLATION
                        1181 ;
                        1182 ; SIDE EFFECTS:
                        1183 ;
                        1184 ;       NONE
                        1185 ;
                        1186 ;--
```

```
                                007C  059D  1188 .ENTRY  EXE$SETPRT,^M<R2,R3,R4,R5,R6>   ; ENTRY POINT
                                      059F  1189
                                      059F  1190          CMK      SETPRT              ; Do set protection                    [20]
                                  04  05AE  1191          RET                          ; Return home                          [20]
                                      05AF  1192
                                      05AF  1193 CMK$SETPRT::
                          50    08 AC   D0  05AF  1194          MOVL     SETPRT$_RETADR(AP), R0          ; INIT THE RETURN ARAY
                                   06   13  05B3  1195          BEQL     5$                              ; IF THERE IS ONE
                             80    00   D2  05B5  1196          MCOML    #0,(R0)+                        ;
                             60    00   D2  05B8  1197          MCOML    #0,(R0)
                                      05BB  1198 5$:
                          50    04 AC   D0  05BB  1199          MOVL     SETPRT$_INADR(AP), R0           ; INPUT ARAY
                       53 80    F7 8F   78  05BF  1200          ASHL     #-9,(R0)+,R3                    ; STARTING VA PFN
                       54 60    F7 8F   78  05C4  1201          ASHL     #-9,(R0),R4                     ; ENDING VA PFN
                          55 54    53   C3  05C9  1202          SUBL3    R3,R4,R5                        ; PAGES MINUS 1
                                   0A   18  05CD  1203          BGEQ     10$                             ; BETTER BE ONE
                    50    00660020 8F   D0  05CF  1204          MOVL     #DS$_PROGERR,R0                 ; INVALID ARGUMENT
                                0031   31  05D6  1205          BRW      SETPRT_XIT                      ; RETURN
                                      05D9  1206 10$:
                          54    08 AC   D0  05D9  1207          MOVL     SETPRT$_RETADR(AP), R4          ; RETURN ADDRESS ARAY
                                55    D6  05DD  1208          INCL     R5                              ; NUMBER OF PAGES
                                      05DF  1209 20$:
                       56 53    02    15   EF  05DF  1210          EXTZV    #21,#2,R3,R6                    ; ADDRESS SPACE INDEX
                             F3'AF   9F  05E4  1211          PUSHAB   B^30$                           ; Setup return address
                                      05E7  1212          CASE     R6,LIMIT=#0,TYPE=L, -
                                      05E7  1213                   DISPLIST=<PZERO,PONE,SYSTEM,RESERVD>
                                      05F3  1214
                             14 50    E9  05F3  1215 30$:     BLBC     R0,SETPRT_XIT                   ; IF ERROR EXIT
                                53    D6  05F6  1216          INCL     R3                              ; INCREMENT THE PFN
                             E4 55    F5  05F8  1217          SOBGTR   R5,20$                          ; DO THE NEXT PAGE
                             08 AC    D5  05FB  1218          TSTL     SETPRT$_RETADR(AP)              ; IS THERE A RETURN ADDRESS ?
                                CA    13  05FE  1219          BEQL     SETPRT_XIT                      ; NO -
08 BC    04 BC    000001FF 8F   CB  0600  1220          BICL3    #^X1FF, @SETPRT$_INADR(AP), -   ; YES-INDICATE THE STARTING     [19]
                                      060A  1221            @SETPRT$_RETADR(AP)             ; VIRTUAL ADDRESS
                                      060A  1222 SETPRT_XIT:
                                   02 060A  1223          REI                          ; Return from exception                [20]
```

```
                                        060B  1225 ;+
                                        060B  1226 ; SET PROTECTION FOR P0 ADDRESS SPACE <3FFFFFFF:00000000>
                                        060B  1227 ; R3 = VIRTUAL PAGE FRAME NUMBER
                                        060B  1228 ; R4 = ENDING RETURN VIRTUAL ADDRESS
                                        060B  1229 ;-
                                        060B  1230 PZERO:
      51    53    15    00    EF    060B  1231          EXTZV   #0,#21,R3,R1            ; PFN <29:9>
                  50    09    DB    0610  1232          MFPR    #PR$_P0LR,R0            ; P0 LENGTH REGISTER
                  51    50    D1    0613  1233          CMPL    R0,R1                   ; VA<29:9> GEQ P0LR ?
                        06    14    0616  1234          BGTR    10$                     ; NO - LENGTH WITHIN RANGE
            50    018C  8F    3C    0618  1235          MOVZWL  #SS$_LENVIO,R0          ; YES- LENGTH FAULT
                        05    10    061B  1236          BSBB    SET                     ; SET THE PROTECTION
                        05    10    061D  1237 10$:     RSB                             ; RETURN
                              05    061E  1238          RSB                             ; RETURN
                                    0620  1239
                                    0621  1240 ;+
                                    0621  1241 ; SET PROTECTION FOR P1 ADDRESS SPACE <7FFFFFFF:40000000>
                                    0621  1242 ; R3 = VIRTUAL PAGE FRAME NUMBER
                                    0621  1243 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                                    0621  1244 ;-
                                    0621  1245 PONE:
      51    53    15    00    EF    0621  1246          EXTZV   #0,#21,R3,R1            ; PFN <29:9>
                  50    08    DB    0626  1247          MFPR    #PR$_P1LR,R0            ; P1 LENGTH REGISTER
                  51    50    D1    0629  1248          CMPL    R0,R1                   ; VA<29:9> LSS P1LR
                        06    15    062C  1249          BLEQ    10$                     ; NO - LENGTH WITHIN RANGE
            50    018C  8F    3C    062E  1250          MOVZWL  #SS$_LENVIO,R0          ; YES- LENGTH VIOLATION
                        05    10    0633  1251          RSB                             ; RETURN
                        1B    10    0634  1252 10$:     BSBB    SET                     ; SET PROTECTION
                              05    0636  1253          RSB                             ; RETURN
                                    0637  1254
                                    0637  1255 ;+
                                    0637  1256 ; SET PROTECTION FOR SYSTEM ADDRESS SPACE <BFFFFFFF:80000000>
                                    0637  1257 ; R3 = VIRTUAL PAGE FRAME NUMBER
                                    0637  1258 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                                    0637  1259 ;-
                                    0637  1260 SYSTEM:
      51    53    15    00    EF    0637  1261          EXTZV   #0,#21,R3,R1            ; PFN <30:9>
                  50    0D    DB    063C  1262          MFPR    #PR$_SLR,R0             ; SYSTEM LENGTH REGISTER
                  51    50    D1    063F  1263          CMPL    R0,R1                   ; VA<30:9> GTR SLR
                        06    14    0642  1264          BGTR    10$                     ; NO - LENGTH WITHIN RANGE
            50    018C  8F    3C    0644  1265          MOVZWL  #SS$_LENVIO,R0          ; YES- LENGTH VIOLATION
                        05    10    0649  1266          BSBB    SET                     ; SET PROTECTION
                        05    10    064A  1267 10$:     RSB                             ; RETURN
                              05    064C  1268          RSB                             ; RETURN
                                    064D  1269
                                    064D  1270 ;+
                                    064D  1271 ; SET PROTECTION FOR RESERVED ADDRESS SPACE <FFFFFFFF:C0000000>
                                    064D  1272 ; THIS FUNCTION IS ILLEGAL
                                    064D  1273 ; R3 = VIRTUAL PAGE FRAME NUMBER
                                    064D  1274 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                                    064D  1275 ;-
                                    064D  1276 RESERVD:
                  50    0C    D0    064D  1277          MOVL    #SS$_ACCVIO,R0          ; INDICATE ACCESS VIOLATION
                              05    0650  1278          RSB                             ; RETURN
                                    0651  1279
                                    0651  1280 ;-
                                    0651  1281 ; SUBROUTINE TO SET PAGE PROTECTION
```

```
                                0651  1282 ; R3 = VIRTUAL PAGE FRAME NUMBER
                                0651  1283 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                                0651  1284 ;-
                                0651  1285 SET:
            51   53   09   78  0651  1286          ASHL    #9,R3,R1                ; MAKE ADDRESS OF PAGE FROM PFN
                      2E   10  0655  1287          BSBB    RPTEADR                 ; GET ADDRESS OF PTE
                 14   AC   D5  0657  1288          TSTL    SETPRT$_PRVPRT(AP)      ; IS THERE A PREVIOUS
                                065A  1289                                          ; PROTECTION ADDRESS?
                      09   13  065A  1290          BEQL    10$                     ; NO -
        50   61   04   1B   EF  065C  1291          EXTZV   #27, #4, (R1), R0       ; Get old protection            [20]
             14   BC   50   90  0661  1292          MOVB    R0, @SETPRT$_PRVPRT(AP) ; Store it                      [20]
                                0665  1293                                          ; YES- PASS TO CALLER
                                0665  1294                                          ; AS PREVIOUS PROTECTION
                                0665  1295 10$:
      61   04   1B   10   AC   F0  C665  1296          INSV    SETPRT$_PROT(AP),#27,#4,(R1) ; INSERT NEW PROTECTION
            51   53   09   78  066B  1297          ASHL    #9, R3, R1              ; FORM ADDRESS
                 3A   51   DA  066F  1298          MTPR    R1, #PR$_TBIS           ; FLUSH TRANSLATION BUF.
              50   08   AC   D0  0672  1299          MOVL    SETPRT$_RETADR(AP),R0   ; IS THERE A RETURN ADDRESS ?
                      09   13  0676  1300          BEQL    20$                     ; NO -
  04 A0   51   00001FFF  8F   C9  0678  1301          BISL3   #^X1FF,R1,4(R0)         ; RETURN HIGHEST ADDRESS OF PAGE
                                0681  1302 20$:
                 50   01   D0  0681  1303          MOVL    #SS$_NORMAL,R0          ; INDICATE SUCCESS
                      05   0684  1304          RSB                             ; RETURN
                                0685  1305
```

```
                                0685  1307          .SBTTL  CALCULATE PTE ADDRESS SUBROUTINE
                                0685  1308 ;++
                                0685  1309 ; RPTEADR - CALCULATE PTE ADDRESS SUBROUTINE
                                0685  1310 ;
                                0685  1311 ; THIS SUBROUTINE IS CALLED TO CALCULATE THE ADDRESS OF A PAGE TABLE
                                0685  1312 ; ENTRY, GIVEN A VIRTUAL ADDRESS
                                0685  1313 ;
                                0685  1314 ; INPUTS:
                                0685  1315 ;
                                0685  1316 ;       R1 = VIRTUAL ADDRESS
                                0685  1317 ;
                                0685  1318 ; OUTPUTS:
                                0685  1319 ;
                                0685  1320 ;       R1 = PTE ADDRESS
                                0685  1321 ;--
                                0685  1322
                                0685  1323 RPTEADR::                                    ;
          02 51   1F   E1       0685  1324          BBC     #31,R1,10$                  ; IF C, P0 OR P1 ADDRESS
                  2C   11       0689  1325          BRB     100$                        ; ELSE, CALCULATE SPTE ADDRESS
          05 51   1E   E0       068B  1326 10$:     BBS     #30,R1,20$                  ; IF S, P1 ADDRESS
             7E   08   DB       068F  1327          MFPR    #PR$_P0BR,-(SP)             ; ELSE, GET P0PT SVA BASE
                  03   11       0692  1328          BRB     30$                         ;  AND PROCEED
             7E   0A   DB       0694  1329 20$:     MFPR    #PR$_P1BR,-(SP)             ; GET P1PT SVA BASE
       51    51   15   09   EF  0697  1330 30$:     EXTZV   #9,#21,R1,R1                ; PVA PFN
             51   51   02   78  069C  1331          ASHL    #2,R1,R1                    ; SPT'PxPT OFFSET
             51   8C   C0   06A0  1332          ADDL2   (SP)+,R1                    ; SVA
       7E    51   09   00   EF  06A3  1333          EXTZV   #0,#9,R1,-(SP)              ; SAVE PxPT OFFSET
                  0D   10       06A8  1334          BSBB    100$                        ; CALCULATE SPTE ADDRESS
       51    61   15   00   EF  06AA  1335          EXTZV   #0,#21,(R1),R1              ; PHYSICAL PxPT ADDRESS
             51   51   09   78  06AF  1336          ASHL    #9,R1,R1                    ; SHIFT TO PFN POSITION
             51   8E   C0   06B3  1337          ADDL2   (SP)+,R1                    ; PxPTE ADDRESS
                       05   06B6  1338          RSB                                 ; RETURN
                            06B7  1339
                            06B7  1340 ;
                            06B7  1341 ; CALCULATE SPTE PHYSICAL ADDRESS
                            06B7  1342 ;
                            06B7  1343
             7E   0C   DB   06B7  1344 100$:     MFPR    #PR$_SBR,-(SP)              ; SPT BASE
       51    51   15   09   EF  06BA  1345          EXTZV   #9,#21,R1,R1                ; SVA PFN
             51   51   02   78  06BF  1346          ASHL    #2,R1,R1                    ; SPT OFFSET
             51   8E   C0   06C3  1347          ADDL2   (SP)+,R1                    ; SPTE ADDRESS
                       05   06C6  1348          RSB                                 ; RETURN
```

```
                        06C7  1350 ;++
                        06C7  1351 ; FUNCTIONAL DESCRIPTION:
                        06C7  1352 ;
                        06C7  1353 ;       These entry points are here only to satify unneeded
                        06C7  1354 ;       functionality.
                        06C7  1355 ;
                        06C7  1356 ;       MMG$IOLOCK      Routine to lock page(s) in memory, returns PTE addr
                        06C7  1357 ;       MMG$UNLOCK      Routine to unlock page(s)
                        06C7  1358 ;       INI$RDONLY      Routine to make read only parts of DS readonly
                        06C7  1359 ;       INI$WRITEABLE   Routine to make DS writeable
                        06C7  1360 ;
                        06C7  1361 ; CALLING SEQUENCE:
                        06C7  1362 ;
                        06C7  1363 ;       BSBW    ???
                        C6C7  1364 ;
                        06C7  1365 ; INPUT PARAMETERS:
                        06C7  1366 ;
                        06C7  1367 ;       Varied but all unused
                        06C7  1368 ;
                        06C7  1369 ; IMPLICIT INPUTS:       NONE
                        06C7  1370 ;
                        06C7  1371 ; OUTPUT PARAMETERS:
                        06C7  1372 ;
                        06C7  1373 ;       MMG$IOLOCK      R1 - Address of PTE for address in R0 on entry
                        06C7  1374 ;
                        06C7  1375 ; IMPLICIT OUTPUTS:      NONE
                        06C7  1376 ;
                        06C7  1377 ; SIDE EFFECTS:          NONE
                        06C7  1378 ;
                        06C7  1379 ; COMPLETION CODES:      SS$_NORMAL - Always
                        06C7  1380 ;--
                        06C7  1381
                        06C7  1382 MMG$IOLOCK::
   51    50    D0       06C7  1383          MOVL    R0,R1                   ; Copy starting virtural address
     B8  AF    16       06CA  1384          Jsb     RPTEADR                 ; Get address of first PTE      [27]
00 51    1F    E2       06CD  1385          BBSS    #31,R1,MMG$UNLOCK       ; Make it a system virtural address also
                        06D1  1386
                        06D1  1387 MMG$UNLOCK::
   50    01    D0       06D1  1388          MOVL    #SS$_NORMAL,R0          ; Set success
                 05       06D4  1389          RSB
```

```
                                06D5  1391 .SBTTL DSX$MAPDBGBLOCK - MAP A BLOCK OF MEMORY FOR THE DEBUGGER          [23]
                                06D5  1392
                                06D5  1393 ;++                                                                    [23]
                                06D5  1394 ; FUNCTIONAL DESCRIPTION                                               [23]
                                06D5  1395 ;       This routine will map a section of memory space and add the    [23]
                                06D5  1396 ; space to the debugger's total memory allocation.  The new space is   [23]
                                06D5  1397 ; allocated just below that which has already been allocated, resulting [23]
                                06D5  1398 ; in contiguous downward growth of memory space assigned to the debug- [23]
                                06D5  1399 ; ger.  This routine is for standalone use only.                       [23]
                                06D5  1400 ;                                                                      [23]
                                06D5  1401 ; INPUTS:                                                              [23]
                                06D5  1402 ;       4(AP) - size of requested memory block                         [23]
                                06D5  1403 ;       8(AP) - address of location to return starting address of      [23]
                                06D5  1404 ;               new buffer space.                                      [23]
                                06D5  1405 ;                                                                      [23]
                                06D5  1406 ; OUTPUTS:                                                             [23]
                                06D5  1407 ;       @8(AP) - starting addr. of new buffer space                    [23]
                                06D5  1408 ;       R0 - 1 if memory space allocated                               [23]
                                06D5  1409 ;            0 if insufficient free memory to allow allocation         [23]
                                06D5  1410 ;--
                                06D5  1411
                         000C   06D5  1412 .ENTRY  DSX$MAPDBGBLOCK, ^M<R2,R3>;                                     [23]
                                06D7  1413
                          1C    E0  06D7  1414          BBS     #DSA$V_USER,-          ;IF USER MODE              [23]
                 43 0000FE00'EF        06D9  1415                  DSA$GL_FLAGS,1000$     ;THEN JUST RETURN          [23]
                      52   04 AC    D0  06DF  1416          MOVL    4(AP),R2               ;GET REQUESTED SIZE        [23]
                52    000001FF 8F    D3  06E3  1417          BITL    #^X1FF,R2              ;IF NOT A PAGE BOUNDARY    [23]
                          0E    13  06EA  1418          BEQL    10$                    ;THEN                      [23]
                      52    000001FF 8F    CA  06EC  1419          BICL    #^X1FF,R2              ;    CLEAR PAGE OFFSET     [23]
                      52    00000200 8F    C0  06F3  1420          ADDL2   #^X200,R2              ;    MAKE IT ONE PAGE BIGGER [23]
            52    00000000'EF    52    C3  06FA  1421 10$:     SUBL3   R2,DEBUG_LO,R2         ;CALC. NEW LOW BUFFER LIMIT [23]
                  53    00000000'EF    D0  0702  1422          MOVL    DEBUG_LO,R3            ;SET TOP OF NEW AREA       [23]
                                52    DD  0709  1423          PUSHL   R2                     ;SAVE R2 FOR LATER         [23]
                        0000078D'EF    16  0708  1424          Jsb     MAP_BLK                ;CALL ROUTINE TO DO THE MAP [27]
                                52 8ED0  0711  1425          POPL    R2                     ;RESTORE LOW BUFFER LIMIT  [23]
                                   08 50    E9  0714  1426          BLBC    R0,1000$               ;IF NO ERROR FROM MAP_BLK  [23]
                        00000000'EF    52    D0  0717  1427          MOVL    R2,DEBUG_LO            ;THEN SET NEW LOW LIMIT    [23]
                                08 BC  52    D0  071E  1428          MOVL    R2,@8(AP)              ;PASS IT TO CALLER ALSO    [23]
                                       04    0722  1429 1000$:  RET                            ;RETURN (R0 EITHER SET OR CLEARED)[23]
                                             0723  1430
```

```
                                0723  1432 .SBTTL DSX$FREEDBGSYM - FREE MEMORY MAPPED TO DEBUGGER SYMTABS
                                0723  1433
                                0723  1434 ;++                                                                    [24]
                                0723  1435 ; FUNCTIONAL DESCRIPTION                                               [24]
                                0723  1436 ;       This routine will deallocate the memory which has been        [24]
                                0723  1437 ; assigned to the debugger's symbol tables.                           [24]
                                0723  1438 ;                                                                     [24]
                                0723  1439 ; INPUTS                                                            [24]
                                0723  1440 ;       none                                                          [24]
                                0723  1441 ;                                                                     [24]
                                0723  1442 ; OUTPUTS                                                           [24]
                                0723  1443 ;       none                                                          [24]
                                0723  1444 ;                                                                     [24]
                                0723  1445 ; IMPLICIT INPUTS                                                    [24]
                                0723  1446 ;       DEBUG_LO - Bottom of debugger's memory allocation, i. e.,     [24]
                                0723  1447 ;                  bottom of symbol table                             [24]
                                0723  1448 ;       SYMTAB_HI- Top of symbol table                                [24]
                                0723  1449 ;--                                                                   [24]
                                0723  1450
                         000C   0723  1451 .ENTRY  DSX$FREEDBGSYM, ^M<R2,R3>       ;                            [24]
                                0725  1452
                   1C    E0     0725  1453         BBS     #DSA$V_USER,-          ;IF USER MODE                [24]
              1F 0000FE00'EF     0727  1454                 DSA$GL_FLAGS,1000$     ;THEN JUST RETURN            [24]
           52    00000000'EF   D0  072D  1455         MOVL    DEBUG_LO,R2           ;GET BOTTOM OF TABLES        [24]
           53    00000000'EF   D0  0734  1456         MOVL    SYMTAB_HI,R3          ;GET TOP OF TABLES           [24]
                 000007E5'EF   16  073B  1457         Jsb     UNMAP_BLK             ;DEALLOCATE THE AREA         [27]
00000000'EF      00000000'EF   D0  0741  1458         MOVL    SYMTAB_HI,DEBUG_LO    ;ADJUST DEBUGGER LOW ADDRESS [24]
                         04    074C  1459 1000$:  RET                             ;THAT'S ALL                  [24]
                               074D  1460
```

```
                                074D  1462 .SBTTL MAP_DEBUGGER - MAP THE DEBUGGER'S MEMORY SPACE                    [23]
                                074D  1463
                                074D  1464 ;++                                                                    [23]
                                074D  1465 ; FUNCTIONAL DESCRIPTION                                               [23]
                                074D  1466 ;       This routine will map all memory allocated to the debugger     [23]
                                074D  1467 ; This includes memory space allocated both for the debugger's         [23]
                                074D  1468 ; executable code and for the buffer space assigned for symbol table   [23]
                                074D  1469 ; construction.                                                        [23]
                                074D  1470 ;                                                                      [23]
                                074D  1471 ; INPUTS                                                               [23]
                                074D  1472 ;       NONE                                                           [23]
                                074D  1473 ;                                                                      [23]
                                074D  1474 ; OUTPUTS                                                              [23]
                                074D  1475 ;       R0 = 1 if space mapped successfully                            [23]
                                074D  1476 ;          = 0 if space already mapped (this may not be an error)      [23]
                                074D  1477 ;--
                                074D  1478
                                074D  1479 MAP_DEBUGGER::                       ;                                 [23]
                          0C    BB  074D  1480          PUSHR   #^M<R2,R3>      ;SAVE REGISTERS                   [23]
          52  00000000'EF     D0  074F  1481          MOVL    DEBUG_LO,R2     ;GET LOW LIMIT OF ALLOCATION      [23]
          53  00000000'EF     D0  0756  1482          MOVL    DEBUG_HI,R3     ;GET HIGH LIMIT OF ALLOCATION     [23]
              0000078D'EF     16  075D  1483          Jsb     MAP_BLK         ;DO THE MAP                       [27]
                          0C    BA  0763  1484          POPR    #^M<R2,R3>      ;RESTORE REGISTERS                [23]
                                05  0765  1485          RSB                     ;RETURN                           [23]
```

```
                                0766  1487 .SBTTL UNMAP_DEBUGGER - UNMAP THE DEBUGGER'S MEMORY SPACE                [23]
                                0766  1488
                                0766  1489 ;++                                                                     [23]
                                0766  1490 ; FUNCTIONAL DESCRIPTION                                                 [23]
                                0766  1491 ;       This routine will unmap the pages of memory previously         [23]
                                0766  1492 ; allocated to the debugger.                                            [23]
                                0766  1493 ;                                                                       [23]
                                0766  1494 ; INPUTS                                                              [23]
                                0766  1495 ;       NONE                                                           [23]
                                0766  1496 ;                                                                       [23]
                                0766  1497 ; OUTPUTS                                                             [23]
                                0766  1498 ;       NONE                                                           [23]
                                0766  1499 ;--                                                                    [23]
                                0766  1500
                                C766  1501 UNMAP_DEBUGGER::                     ;                              [23]
                          0C    BB  0766  1502          PUSHR   #^M<R2,R3>      ;SAVE REGISTERS                [23]
52  00000000'EF               D0  0768  1503          MOVL    DEBUG_LO,R2     ;GET LOW LIMIT OF ALLOCATION   [23]
52  000001FF 8F               CA  076F  1504          BICL2   #^X1FF,R2       ;MAKE SURE IT'S A PAGE BOUNDARY [23]
53  00000000'EF               D0  0776  1505          MOVL    DEBUG_HI,R3     ;GET HIGH LIMIT OF ALLOCATION  [23]
53  000001FF 8F               CA  077D  1506          BICL2   #^X1FF,R3       ;MAKE SURE IT'S A PAGE BOUNDARY [23]
    000007E5'EF               16  0784  1507          Jsb     UNMAP_BLK       ;DO THE MAP                    [27]
                          0C    BA  078A  1508          POPR    #^M<R2,R3>      ;RESTORE REGISTERS             [23]
                                05  078C  1509          RSB                     ;RETURN                        [23]
                                    078D  1510
```

```
                                078D  1512 ; ROUTINE TO MAP A SPECIFIED BLOCK OF MEMORY.  IF THE BLOCK HAS ALREADY [23]
                                078D  1513 ; BEEN MAPPED, INDICATE ERROR RETURN.                                    [23]
                                078D  1514 ;                                                                        [23]
                                078D  1515 ; INPUTS                                                                 [23]
                                078D  1516 ;       R2 = BOTTOM ADDR. OF NEW BUFFER                                  [23]
                                078D  1517 ;       R3 = TOP ADDR. OF NEW BUFFER                                     [23]
                                078D  1518 ;                                                                        [23]
                                078D  1519 ; OUTPUTS                                                                [23]
                                078D  1520 ;       R0 = 0 IF SPACE MAPPED SUCCESSFULLY                              [23]
                                078D  1521 ;          = 1 IF SPACE ALREADY ALLOCATED                                [23]
                                078D  1522 ;                                                                        [23]
                                078D  1523
                                078D  1524 MAP_BLK:
                          54    DD  078D  1525         PUSHL   R4                      ;SAVE A REGISTER             [23]
                    54    52    D0  078F  1526         MOVL    R2,R4                   ;SAVE IT FOR LATER           [23]
                                0792  1527                                            ;NOW TEST REQUESTED BUFFER AREA [23]
                                0792  1528                                            ;TO MAKE SURE IT HASN'T ALREADY [23]
                                0792  1529                                            ;BEEN ALLOCATED TO SOMEONE ELSE [23]
                                0792  1530 100$:                                      ;REPEAT                      [23]
                    51    52    D0  0792  1531         MOVL    R2,R1                   ;   GET NEXT PAGE           [23]
                  FEEC    CF    16  0795  1532         Jsb     RPTEADR                 ;   GET PTE OF PAGE         [27]
              43  61    1F    E0  0799  1533         BBS     #PTE$V_VALID,(R1),1000$ ;   IF PAGE ALLOCATED, LEAVE [23]
          52  00000200  8F    C0  079D  1534         ADDL2   #512,R2                 ;   ADDR. OF NEXT PAGE      [23]
                    53    52    D1  07A4  1535         CMPL    R2,R3                   ;   SEE IF TOP OF NEW BUFFER AREA[23]
                          E9    19  07A7  1536         BLSS    100$                    ;UNTIL ENTIRE NEW BUFFER CHECKED[23]
                                07A9  1537                                            ;IT'S OK TO ALLOCATE THE SPACE [23]
                    52    54    D0  07A9  1538         MOVL    R4,R2                   ;GET BOTTOM OF BUFFER        [23]
                                07AC  1539 200$:                                      ;REPEAT                      [23]
                    51    52    D0  07AC  1540         MOVL    R2,R1                   ;   GET NEXT PAGE           [23]
                  FED2    CF    16  07AF  1541         Jsb     RPTEADR                 ;   GET PTE OF PAGE         [27]
              54  01    1F    78  07B3  1542         ASHL    #PTE$V_VALID,#1,R4      ;   CREATE BIT MASK - SET VALID [23]
          54  02    17    00  F0  07B7  1543         INSV    #0,#PTE$V_OWN,#PTE$S_OWN,R4;SET OWNER BITS TO ZERO   [23]
           54   20000000  8F    C8  07BC  1544         BISL2   #PTE$C_UW,R4            ;   SET USER-WRITABLE IN MASK [23]
           61   78000000  8F    CA  07C3  1545         BICL2   #PTE$M_PROT,(R1)        ;   CLEAR PROTECTION IN PTE [23]
                      61    54    C8  07CA  1546         BISL2   R4,(R1)             ;   SET BITS IN PTE         [23]
          52  00000200  8F    C0  07CD  1547         ADDL2   #512,R2                 ;   ADDR. OF NEXT PAGE      [23]
                    53    52    D1  07D4  1548         CMPL    R2,R3                   ;   SEE IF TOP OF NEW BUFFER AREA[23]
                          D3    19  07D7  1549         BLSS    200$                    ;UNTIL ENTIRE NEW BUFFER MAPPED [23]
                    50    01    D0  07D9  1550         MOVL    #1,R0                   ;SET SUCCESS                 [23]
                                07DC  1551 900$:       POPL    R4                      ;RESTORE A REGISTER          [23]
                          05    07DF  1552             RSB                             ;RETURN                      [23]
                    50    00    D0  07E0  1553 1000$:  MOVL    #0,R0                   ;SET ERROR                   [23]
                          F7    11  07E3  1554         BRB     900$                    ;RETURN                      [23]
                                07E5  1555
                                07E5  1556
                                07E5  1557 ; ROUTINE TO UNMAP A SPECIFIED BLOCK OF MEMORY.                          [23]
                                07E5  1558 ;                                                                        [23]
                                07E5  1559 ; INPUTS:                                                                [23]
                                07E5  1560 ;       R2 = BOTTOM ADDR. OF AREA TO UNMAP                               [23]
                                07E5  1561 ;       R3 = TOP ADDR. OF AREA TO UNMAP                                  [23]
                                07E5  1562 ;                                                                        [23]
                                07E5  1563 ; OUTPUTS:                                                               [23]
                                07E5  1564 ;       NONE                                                             [23]
                                07E5  1565 ;                                                                        [23]
                                07E5  1566
                                07E5  1567 UNMAP_BLK:                                 ;                             [23]
                                07E5  1568 100$:                                      ;REPEAT                      [23]
```

```
                 51    52   D0  07E5  1569          MOVL    R2,R1                           ;    GET A PAGE                        [23]
                 FE99  CF   16  07E8  1570          Jsb     RPTEADR                         ;    GET ADDR. OF PTE                  [27]
          61     81800000 8F   CA  07EC  1571          BICL2   #PTE$M_VALID+PTE$M_OWN,(R1);MAKE PAGE INVALID                  [23]
          52     00000200 8F   C0  07F3  1572          ADDL2   #512,R2                        ;    GET NEXT PAGE                     [23]
                   53    52   D1  07FA  1573          CMPL    R2,R3                           ;    SEE IF TOP OF AREA YET            [23]
                         E6   19  07FD  1574          BLSS    100$                            ;UNTIL TOP OF AREA REACHED             [23]
                              05  07FF  1575          RSB                                     ;RETURN                                [23]
                                  0800  1576
                                  0800  1577 ; Referenced by XDELTA
                                  0800  1578 ;
                                  0800  1579 INI$RDONLY::
                                  0800  1580 INI$WRITABLE::
                              05  0800  1581          RSB
                                  0801  1582
                                  0801  1583          .END
```

```
$$ARGS                          =  00000005     D
$$N                             =  00000002     D
$$T1                            =  00000018     D
$ER                             =  00000002     D
$MODULE                            00000000 R   D   03
ACCESSABLE                         00000207 R   D   04
A_P1BUFPTE                         00000004 R   D   02
A_P1BUFVA                          00000008 R   D   02
A_SPTEND                           0000000C R   D   02
BIT...                          =  00000015     D
BIT0                            =  00000001     D
BIT1                            =  00000002     D
BIT10                           =  00000400     D
BIT11                           =  00000800     D
BIT12                           =  00001000     D
BIT13                           =  00002000     D
BIT14                           =  00004000     D
BIT15                           =  00008000     D
BIT16                           =  00010000     D
BIT17                           =  00020000     D
BIT18                           =  00040000     D
BIT19                           =  00080000     D
BIT2                            =  00000004     D
BIT20                           =  00100000     D
BIT21                           =  00200000     D
BIT22                           =  00400000     D
BIT23                           =  00800000     D
BIT24                           =  01000000     D
BIT25                           =  02000000     D
BIT26                           =  04000000     D
BIT27                           =  08000000     D
BIT28                           =  10000000     D
BIT29                           =  20000000     D
BIT3                            =  00000008     D
BIT30                           =  40000000     D
BIT31                           =  80000000     D
BIT4                            =  00000010     D
BIT5                            =  00000020     D
BIT6                            =  00000040     D
BIT7                            =  00000080     D
BIT8                            =  00000100     D
BIT9                            =  00000200     D
CEP_FUNCTIONAL                  =  00000000     D
CEP_REPAIR                      =  00000001     D
CF$L_AP                            00000008     D
CF$L_FP                            0000000C     D
CF$L_ONCOND                        00000000     D
CF$L_PC                            00000010     D
CF$L_REG                           00000014     D
CF$W_MASK                          00000006     D
CF$W_PSW                           00000004     D
CMK$_MMENABLE                      0000058C RG  D   04
CMK$_SETPRT                        000005AF RG  D   04
CMK$_                           =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
CNTREG$_ACMODE                  =  0000000C     D
CNTREG$_NARGS                   =  00000004     D
CNTREG$_PAGCNT                  =  00000004     D
CNTREG$_REGION                  =  00000010     D
CNTREG$_RETADR                  =  00000008     D
CRD$EXPREG                         00000346 RG  D   04
DEBUG_HI                           ********     X   00
DEBUG_LO                           ********     X   00
DS$AQ_SYSSRV                       ********     X   00
DS$A_PRGBGN                        ********     X   00
DS$GA_LASTADR                      ********     X   00
DS$GB_MM_ENB                       0000001C RG  D   02
DS$GL_ACTUAL_MEMSIZE               00000014 RG  D   02
DS$GL_BUFCNT                       ********     X   00
DS$GL_FLAGS                        ********     X   00
DS$GL_MEMSIZE                      00000010 RG  D   02
DS$GL_SET_MEMSIZE                  00000018 RG  D   02
DS$GQ_PHYADR                       ********     X   00
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_PRGSIZ                        ********     X   00
DS$K_PRINTD                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  0000001B     D
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        =  0000000B     D
DS$K_TYPE_FIRST_PASS            =  00000011     D
```

```
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INSFMEM                     = 00660050     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
```

```
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_USER                  =     0000001C     D
DSR$CHECKLOAD                     ********   X   00
DSR$MMENABLE                      0000057C RG  D   04
DSR$SETIMR_INI                    ********   X   00
DSX$FREEDBGSYM                    00000723 RG  D   04
DSX$MAPDBGBLOCK                   000006D5 RG  D   04
DSX$MMOFF                         00000562 RG  D   04
DSX$MMOFF_X                       00000578 R   D   04
DSX$MMON                          00000554 RG  D   04
DSX$MMON_X                        00000561 R   D   04
DSX$PRINT                         ********   X   00
DS_ERRSUP                         ********   X   00
ENV$M_DOMAIN                =     00000002     D
ENV$M_LEVEL                 =     00000001     D
ENV$M_SUPER                 =     000003FC     D
ENV$S_DOMAIN                =     00000001     D
ENV$S_LEVEL                 =     00000001     D
ENV$S_SUPER                 =     00000008     D
ENV$V_DOMAIN                =     00000001     D
ENV$V_LEVEL                 =     00000000     D
ENV$V_SUPER                 =     00000002     D
ENV$_CPU                    =     (  000000    D
ENV$_FUNCTIONAL             =     00000000     D
ENV$_REPAIR                 =     00000001     D
ENV$_SUPER                  =     00000001     D
ENV$_SYSTEM                 =     00000001     D
ESTKPTR                           ********   X   00
EXE$CNTREG                        00000467 RG  D   04
EXE$CNTREG_X                      00000553 R   D   04
EXE$EXPREG                        00000363 RG  D   04
EXE$EXPREG_X                      00000466 R   D   04
EXE$SETPRT                        0000059D RG  D   04
EXPREG$_ACMODE              =     0000000C     D
EXPREG$_NARGS               =     00000004     D
EXPREG$_PAGCNT              =     00000004     D
EXPREG$_REGION              =     00000010     D
EXPREG$_RETADR              =     00000008     D
EXPREG_COMMON                     00000373 R   D   04
GETBUF$_NARGS               =     00000004     D
GETBUF$_PAGCNT              =     00000004     D
GETBUF$_PHYADR              =     0000000C     D
GETBUF$_REGION              =     00000010     D
GETBUF$_RETADR              =     00000008     D
GETMEM$_NARGS               =     00000003     D
GETMEM$_PAGCNT              =     00000004     D
GETMEM$_PHYADR              =     00000008     D
GETMEM$_VRTADR              =     0000000C     D
INI$RDONLY                        00000800 RG  D   04
INI$WRITABLE                      00000800 RG  D   04
IO$GQ_PHYSICAL                    ********   X   00
IRP$B_CARCON                      00000038     D
IRP$B_EFN                         00000022     D
IRP$B_PRI                         00000023     D
IRP$B_RMOD                        0000000B     D
IRP$B_TYPE                        0000000A     D
IRP$C_LENGTH                      0000005C     D
IRP$K_LENGTH                      0000005C     D
IRP$L_ARB                         00000050     D
IRP$L_AST                         00000010     D
IRP$L_ASTPRM                      00000014     D
IRP$L_DIAGBUF                     00000044     D
IRP$L_EXTEND                      0000004C     D
IRP$L_IOQBL                       00000004     D
IRP$L_IOQFL                       00000000     D
IRP$L_IOSB                        00000024     D
IRP$L_IOST1                       00000034     D
IRP$L_IOST2                       00000038     D
IRP$L_MEDIA                       00000034     D
IRP$L_PID                         0000000C     D
IRP$L_SEGVBN                      00000040     D
IRP$L_SEQNUM                      00000048     D
IRP$L_SVAPTE                      0000002C     D
IRP$L_TT_TERM                     00000038     D
IRP$L_UCB                         0000001C     D
IRP$L_WIND                        00000018     D
IRP$Q_NT_PRVMSK                   0000003C     D
IRP$W_ABCNT                       0000003C     D
IRP$W_BCNT                        00000032     D
IRP$W_BOFF                        00000030     D
IRP$W_CHAN                        00000028     D
IRP$W_FUNC                        00000020     D
IRP$W_OBCNT                       0000003E     D
IRP$W_SIZE                        00000008     D
IRP$W_STS                         0000002A     D
IRP$W_TT_PRMPT                    0000003C     D
ISTKPTR                           ********   X   00
KSTKPTR                           ********   X   00
K_P1PAGES                   =     0000007E     D
L$A_CCP                           00000240     D
L$A_DEVP                          0000021C     D
L$A_DREG                          00000224     D
L$A_DTP                           00000218     D
L$A_ICP                           0000023C     D
L$A_LASTADR                       00000214     D
L$A_NAME                          00000208     D
L$A_REPP                          00000244     D
L$A_SECNAM                        00000250     D
```

```
L$A_STATAB                      00000248     D
L$A_TSTCNT                      00000254     D
L$L_ENVIRON                     00000204     D
L$L_ERRTYP                      0000024C     D
L$L_HEADLENGTH                  00000200     D
L$L_REV                         0000020C     D
L$L_UNIT                        00000220     D
L$L_UNUSED                      00000228     D
L$L_UPDATE                      00000210     D
L_POLEN                         00000000 R   D   02
MAP$IOSPACE                     ********     X   00
MAPFREE                         000002AA RG  D   04
MAPMEM                          00000000 RG  D   04
MAPMEM$IOSPACE                  000001AF RG  D   04
MAP_BLK                         0000C78D R   D   04
MAP_DEBUGGER                    0000074D RG  D   04
MEMPOOL$INIT                    ********     X   00
MMG$GL_SPTBASE                  ********     X   00
MMG$IOLOCK                      000006C7 RG  D   04
MMG$UNLOCK                      000006D1 RG  D   04
POPT_BASE                       ********     X   00
PFN$AB_STATE                =   00000010 RG  D   02
PFN$AB_TYPE                 =   00000000  G  D
PFN$AL_BAK                  =   00000008 RG  D   02
PFN$AL_PTE                  =   00000004 RG  D   02
PFN$AW_BLINK                =   00000000  G  D
PFN$AW_FLINK                =   00000000  G  D
PFN$AW_REFCNT               =   0000000C RG  D   02
PFN$AW_SWPVBN               =   00000000 RG  D   02
PONE                            00000621 R   D   04
PR$_MAPEN                   =   00000038     D
PR$_P0BR                    =   00000008     D
PR$_P0LR                    =   00000009     D
PR$_P1BR                    =   0000000A     D
PR$_P1LR                    =   0000000B     D
PR$_SBR                     =   0000000C     D
PR$_SLR                     =   0000000D     D
PR$_TBIA                    =   00000039     D
PR$_TBIS                    =   0000003A     D
PSL$V_IS                    =   0000001A     D
PTE$C_NA                    =   00000000     D
PTE$C_UREW                  =   60000000     D
PTE$C_URKW                  =   70000000     D
PTE$C_URSW                  =   60000000     D
PTE$C_UW                    =   20000000     D
PTE$M_IO                    =   00100000     D
PTE$M_MODIFY                =   04000000     D
PTE$M_OWN                   =   01800000     D
PTE$M_PFN                   =   001FFFFF     D
PTE$M_PROT                  =   78000000     D
PTE$M_VALID                 =   80000000     D
PTE$S_OWN                   =   00000002     D
PTE$V_IO                    =   00000014     D
PTE$V_OWN                   =   00000017     D
PTE$V_VALID                 =   0000001F     D
PZERO                           0000060B R   D   04
RELBUF$_NARGS               =   00000003     D
RELBUF$_PAGCNT              =   00000004     D
RELBUF$_REGION              =   0000000C     D
RELBUF$_RETADR              =   00000008     D
RELMEM$_NARGS               =   00000003     D
RELMEM$_PAGCNT              =   00000004     D
RELMEM$_PHYADR              =   00000008     D
RELMEM$_VRTADR              =   0000000C     D
RESERVD                         0000064D R   D   04
RPTEADR                         00000685 RG  D   04
SCB$L_ACCESS                    00000020     D
SCB$L_ARITH                     00000034     D
SCB$L_BREAK                     0000002C     D
SCB$L_CHME                      00000044     D
SCB$L_CHMK                      00000040     D
SCB$L_CHMS                      00000048     D
SCB$L_CHMU                      0000004C     D
SCB$L_COMPAT                    00000030     D
SCB$L_KNLSTK                    00000008     D
SCB$L_MACHCHK                   00000004     D
SCB$L_OPCCUS                    00000014     D
SCB$L_OPCDEC                    00000010     D
SCB$L_POWER                     0000000C     D
SCB$L_RADRMOD                   0000001C     D
SCB$L_ROPRAND                   00000018     D
SCB$L_RXDB                      000000F8     D
SCB$L_SFTLVL1                   00000084     D
SCB$L_SFTLVL10                  000000A8     D
SCB$L_SFTLVL11                  000000AC     D
SCB$L_SFTLVL12                  000000B0     D
SCB$L_SFTLVL13                  000000B4     D
SCB$L_SFTLVL14                  000000B8     D
SCB$L_SFTLVL15                  000000BC     D
SCB$L_SFTLVL2                   00000088     D
SCB$L_SFTLVL3                   0000008C     D
SCB$L_SFTLVL4                   00000090     D
SCB$L_SFTLVL5                   00000094     D
SCB$L_SFTLVL6                   00000098     D
SCB$L_SFTLVL7                   0000009C     D
SCB$L_SFTLVL8                   000000A0     D
SCB$L_SFTLVL9                   000000A4     D
SCB$L_TBIT                      00000028     D
SCB$L_TIMER                     000000C0     D
SCB$L_TRANSL                    00000024     D
SCB$L_TXDB                      000000FC     D
SCB$L_ZERO                      00000000     D
SCB_UNKINT                      ********     X   00
SEP_FUNCTIONAL              =   00000002     D
SEP_REPAIR                  =   00000003     D
SET                             00000651 R   D   04
SETPRT$_ACMODE              =   0000000C     D
SETPRT$_INADR               =   00000004     D
SETPRT$_NARGS               =   00000005     D
SETPRT$_PROT                =   00000010     D
SETPRT$_PRVPRT              =   00000014     D
SETPRT$_RETADR              =   00000008     D
SETPRT_XIT                      0000060A R   D   04
SIZ...                      =   00000001     D
```

```
SS$_ACCVIO                      = 0000000C     D
SS$_ILLPAGCNT                   = 000000FC     D
SS$_LENVIO                      = 0000018C     D
SS$_NORMAL                      = 00000001     D
SS$_PAGOWNVIO                   = 000001EC     D
SS$_RESIGNAL                    = 00000918     D
SS$_VASFULL                     = 00000244     D
SSTKPTR                           ********   X   00
STACKPROT                         00000242 R   D   04
STACK_BASE                        ********   X   00
SYMTAB_HI                         ********   X   00
SYS$UNWIND                        ********   X   00
SYSTEM                            00000637 R   D   04
T_INSFMEM                         00000007 R   D   03
UNMAP_BLK                         0000C7E5 R   D   04
UNMAP_DEBUGGER                    00000766 RG  D   04
USTKPTR                           ********   X   00
XDS$INIT                        = 00000000   G  D
```

Psect synopsis

PSECT name	Allocation		PSECT No.	Attributes
. ABS .	00000000	(0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70	(65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000001D	(29.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	0000003E	(62.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000801	(2049.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000005	194 (1)	172 (1) 178 (1) 179 (1) 181 (1) 182 (1) 186 (1) 194 (1)
$$N	=00000002	343 (1)	343 (1)
$$T1	=00000018	194 (1)	172 (1) 178 (1) 179 (1) 181 (1) 182 (1) 186 (1) 194 (1)
$ER	=00000002	973 (12)	#-973 (12)
$MODULE	00000000-R	248 (1)	973 (12)
ACCESSABLE	00000207-R	548 (3)	309 (1)
A_P1BUFPTE	00000004-R	218 (1)	236 (1) #-448 (1) #-694 (6) #-816 (8) #-942 (11)
A_P1BUFVA	00000008-R	219 (1)	237 (1) #-439 (1) #-446 (1) #-818 (8) #-944 (11)
A_SPTEND	0000000C-R	220 (1)	238 (1) #-428 (1) #-709 (6) #-822 (8) #-947 (11)
BIT...	=00000015	193 (1)	169 (1) 170 (1) 176 (1) 177 (1) 184 (1) 193 (1)
BIT0	=00000001	173 (1)	
BIT1	=00000002	173 (1)	
BIT10	=00000400	173 (1)	
BIT11	=00000800	173 (1)	
BIT12	=00001000	173 (1)	
BIT13	=00002000	173 (1)	
BIT14	=00004000	173 (1)	
BIT15	=00008000	173 (1)	
BIT16	=00010000	173 (1)	
BIT17	=00020000	173 (1)	
BIT18	=00040000	173 (1)	
BIT19	=00080000	173 (1)	
BIT2	=00000004	173 (1)	
BIT20	=00100000	173 (1)	
BIT21	=00200000	173 (1)	
BIT22	=00400000	173 (1)	
BIT23	=00800000	173 (1)	
BIT24	=01000000	173 (1)	
BIT25	=02000000	173 (1)	
BIT26	=04000000	173 (1)	
BIT27	=08000000	173 (1)	
BIT28	=10000000	173 (1)	
BIT29	=20000000	173 (1)	
BIT3	=00000008	173 (1)	
BIT30	=40000000	173 (1)	
BIT31	=80000000	173 (1)	
BIT4	=00000010	173 (1)	
BIT5	=00000020	173 (1)	
BIT6	=00000040	173 (1)	
BIT7	=00000080	173 (1)	
BIT8	=00000100	173 (1)	
BIT9	=00000200	173 (1)	
CEP_FUNCTIONAL	=00000000	177 (1)	
CEP_REPAIR	=00000001	177 (1)	

```
CMK$MMENABLE                    0000058C-R   1139  (15)   #-1136  (15)
CMK$SETPRT                      000005AF-R   1193  (17)   #-1190  (17)
CMK$_                          =00000004     169   (1)
CMK$_ASTEXIT                   =00000000     169   (1)
CMK$_CANTIM                    =0000000B     169   (1)
CMK$_HIBER                     =00000007     169   (1)
CMK$_INITSCB                   =00000008     169   (1)
CMK$_LAST                      =0000000E     169   (1)
CMK$_MMENABLE                  =00000003     169   (1)   #-1136  (15)
CMK$_RCONSOLE                  =00000001     169   (1)
CMK$_REFRESH                   =00000005     169   (1)
CMK$_SCHDWK                    =00000006     169   (1)
CMK$_SETIMR                    =0000000C     169   (1)
CMK$_SETIPL                    =00000009     169   (1)
CMK$_SETPRT                    =00000C0D     169   (1)   #-1190  (17)
CMK$_SGIPR                     =0000000A     169   (1)
CMK$_TCONSOLE                  =00000002     169   (1)
CNTREG$_ACMODE                 =0000000C     172   (1)
CNTREG$_NARGS                  =00000004     172   (1)
CNTREG$_PAGCNT                 =00000004     172   (1)   #-983   (12)
CNTREG$_REGION                 =00000010     172   (1)   #-927   (11)  #-934   (11)  #-960   (12)
CNTREG$_RETADR                 =00000008     172   (1)   #-1000  (12)  #-996   (12)
CRD$EXPREG                      00000346-R   775   (8)
DEBUG_HI                        00000000-XR           #-1482  (23)  #-1505  (24)    206   (1)
DEBUG_LO                        00000000-XR           #-1421  (21)  #-1422  (21)  #-1427  (21)  #-1455  (22)
                                                      #-1458  (22)  #-1481  (23)  #-1503  (24)    206   (1)
DS$AQ_SYSSRV                    00000000-XR             200   (1)  #-380   (1)
DS$A_PRGBGN                     00000000-XR             205   (1)    666   (6)
DS$GA_LASTADR                   00000000-XR             200   (1)  #-450   (1)  #-679   (6)  #-777   (8)
DS$GB_MM_ENB                    0000001C-R   229   (1)  #-1098  (14)
DS$GL_ACTUAL_MEMSIZE            00000014-R   223   (1)  #-321   (1)
DS$GL_BUFCNT                    00000000-XR             201   (1)    710   (6)
DS$GL_FLAGS                     00000000-XR             205   (1)
DS$GL_MEMSIZE                   00000010-R   221   (1)   241   (1)  #-335   (1)  #-390   (1)
DS$GL_SET_MEMSIZE               00000018-R   226   (1)  #-322   (1)
DS$GQ_PHYADR                    00000000-XR             202   (1)  #-802   (8)  #-852   (9)  #-853   (9)
DS$K_ERROR                     =00000002     176   (1)
DS$K_NORMAL                    =00000001     176   (1)
DS$K_PRGSIZ                     00000000-XR             205   (1)    666   (6)
DS$K_PRINTB                    =00000002     184   (1)
DS$K_PRINTF                    =00000001     184   (1)  #-343   (1)
DS$K_PRINTI                    =00000000     184   (1)
DS$K_PRINTX                    =00000003     184   (1)
DS$K_SEVERE                    =00000004     176   (1)
DS$K_SUBSYS                    =00000066     176   (1)    176   (1)
DS$K_TYPE_ABORT_PROGRAM        =00000014     184   (1)
DS$K_TYPE_ABORT_TEST           =00000013     184   (1)
DS$K_TYPE_COMMAND_ERR          =00000015     184   (1)
DS$K_TYPE_COMMAND_OUT          =00000016     184   (1)
DS$K_TYPE_CRD_AUTOTEST         =0000001A     184   (1)
DS$K_TYPE_DS_PROMPT            =00000001     184   (1)
DS$K_TYPE_DS_START             =0000001D     184   (1)
DS$K_TYPE_ERRDEV               =00000008     184   (1)
DS$K_TYPE_ERRHARD              =00000006     184   (1)
DS$K_TYPE_ERROR_BODY           =00000009     184   (1)
DS$K_TYPE_ERROR_END            =0000000A     184   (1)
DS$K_TYPE_ERRPREP              =0000001B     184   (1)
```

```
DS$K_TYPE_ERRSOFT               =00000007      184     (1)
DS$K_TYPE_ERRSUP                =00000004      184     (1)
DS$K_TYPE_ERRSYS                =00000005      184     (1)
DS$K_TYPE_ERR_HALT              =0000000D      184     (1)
DS$K_TYPE_EXCEPTION             =0000000C      184     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      184     (1)
DS$K_TYPE_FIRST_PASS            =00000011      184     (1)
DS$K_TYPE_GENERAL               =00000000      184     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      184     (1)     #-343     (1)
DS$K_TYPE_NO_TESTS              =00000012      184     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      184     (1)
DS$K_TYPE_PROGRAM_END           =00000010      184     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      184     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      184     (1)
DS$K_TYPE_QIO_INVADP            =00000C24      184     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      184     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      184     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      184     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      184     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      184     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      184     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      184     (1)
DS$K_TYPE_START_ERR             =00000018      184     (1)
DS$K_TYPE_START_LIST            =00000025      184     (1)
DS$K_TYPE_SUMMARY               =0000000E      184     (1)
DS$K_TYPE_USER_PROMPT           =00000002      184     (1)
DS$K_WARNING                    =00000000      176     (1)
DS$M_ABRTFLG                    =00000040      170     (1)
DS$M_BADTIME                    =00100000      170     (1)
DS$M_BATCH                      =00400000      170     (1)
DS$M_BRKCLR                     =00001000      170     (1)
DS$M_BRKPT                      =00000800      170     (1)
DS$M_CHARFLG                    =00000100      170     (1)
DS$M_CMDFLG                     =00000080      170     (1)
DS$M_CTRLC                      =00000001      170     (1)
DS$M_CTRLO                      =00010000      170     (1)
DS$M_DEVFLG                     =00000200      170     (1)
DS$M_DISABLCC                   =01000000      170     (1)
DS$M_DONFLG                     =00002000      170     (1)
DS$M_ERRFLG                     =00000010      170     (1)
DS$M_EXCEPT                     =00080000      170     (1)
DS$M_EXETST                     =00040000      170     (1)
DS$M_HLTFLG                     =00000008      170     (1)
DS$M_LODFLG                     =00000002      170     (1)
DS$M_MEMMGT                     =00008000      170     (1)
DS$M_OUTPUT                     =00800000      170     (1)
DS$M_RUBFLG                     =00000020      170     (1)
DS$M_SCRIPT                     =00200000      170     (1)
DS$M_SETIMR                     =02000000      170     (1)
DS$M_STRFLG                     =00000004      170     (1)
DS$M_SUBT                       =00004000      170     (1)
DS$M_SYSFLG                     =00000400      170     (1)
DS$M_TIMRON                     =00020000      170     (1)
DS$V_ABRTFLG                    =00000006      170     (1)
DS$V_BADTIME                    =00000014      170     (1)
DS$V_BATCH                      =00000016      170     (1)
DS$V_BRKCLR                     =0000000C      170     (1)
```

```
DS$V_BRKPT                      =0000000B       170     (1)
DS$V_CHARFLG                    =00000008       170     (1)
DS$V_CMDFLG                     =00000007       170     (1)
DS$V_CTRLC                      =00000000       170     (1)
DS$V_CTRLO                      =00000010       170     (1)
DS$V_DEVFLG                     =00000009       170     (1)
DS$V_DISABLCC                   =00000018       170     (1)
DS$V_DONFLG                     =0000000D       170     (1)
DS$V_ERRFLG                     =00000004       170     (1)
DS$V_EXCEPT                     =00000013       170     (1)
DS$V_EXETST                     =00000012       170     (1)
DS$V_HLTFLG                     =00000003       170     (1)
DS$V_LODFLG                     =00000001       170     (1)
DS$V_MEMMGT                     =0000000F       170     (1)
DS$V_OUTPUT                     =00000C17       170     (1)
DS$V_RUBFLG                     =00000005       170     (1)
DS$V_SCRIPT                     =00000015       170     (1)
DS$V_SETIMR                     =00000019       170     (1)
DS$V_STRFLG                     =00000002       170     (1)
DS$V_SUBT                       =0000000E       170     (1)
DS$V_SYSFLG                     =0000000A       170     (1)
DS$V_TIMRON                     =00000011       170     (1)
DS$_ARITH                       =006600D0       176     (1)
DS$_ASBE                        =00660118       176     (1)
DS$_BADLINK                     =006600F0       176     (1)
DS$_BADTYPE                     =006600E8       176     (1)
DS$_BIIC                        =00660120       176     (1)
DS$_CHME                        =006600A8       176     (1)
DS$_CHMK                        =006600E0       176     (1)
DS$_DEVNAME                     =00660108       176     (1)
DS$_ERROR                       =00660002       176     (1)
DS$_IHWE                        =00660068       176     (1)
DS$_FRAGBUF                     =00660080       176     (1)     #-988    (12)
DS$_ICBUSY                      =006600C8       176     (1)
DS$_ICERR                       =006600C0       176     (1)
DS$_IHWE                        =00660060       176     (1)
DS$_ILLCHAR                     =00660018       176     (1)
DS$_ILLPAGCNT                   =00660078       176     (1)
DS$_ILLUNIT                     =00660100       176     (1)
DS$_INSFMEM                     =00660050       176     (1)
DS$_IPL2HI                      =006600B8       176     (1)
DS$_IVADDR                      =00660040       176     (1)
DS$_IVVECT                      =00660038       176     (1)
DS$_KRNLSTK                     =00660090       176     (1)
DS$_LOGIC                       =00660070       176     (1)
DS$_MCHK                        =00660088       176     (1)     #-559    (3)
DS$_MMOFF                       =00660058       176     (1)
DS$_NEEDUNIT                    =006600F8       176     (1)
DS$_NODE                        =00660128       176     (1)
DS$_NOPCS                       =00660110       176     (1)
DS$_NORMAL                      =00660001       176     (1)
DS$_NOSUPPORT                   =006600B1       176     (1)
DS$_NOTDON                      =00660030       176     (1)
DS$_NOTIMP                      =006600B0       176     (1)       176    (1)
DS$_NULLSTR                     =00660010       176     (1)
DS$_OVERFLOW                    =00660008       176     (1)
DS$_POWER                       =00660098       176     (1)
```

```
DS$_PROGERR                     =00660020      176     (1)     #-1204    (17)
DS$_SEVERE                      =00660004      176     (1)
DS$_TRANSL                      =006600A0      176     (1)
DS$_TRUNCATE                    =00660028      176     (1)
DS$_UNEXPINT                    =006600D8      176     (1)
DS$_VASFULL                     =00660048      176     (1)
DS$_WARNING                     =00660000      176     (1)     #-1097    (14)      176     (1)
DSA$AT_APTTXT                    0000FA00                       #-677     (6)
DSA$GL_FLAGS                     0000FE00                         1415    (21)      1454    (22)
DSA$V_USER                      =0000001C                       #-1414    (21)    #-1453    (22)
DSR$CHECKLOAD                    00000000-XR                      208     (1)       667     (6)
DSR$MMENABLE                     0000057C-R     1135    (15)    #-1048    (13)    #-1100    (14)
DSR$SETIMR_INI                   00000000-XR                      201     (1)
DSX$FREEDBGSYM                   00000723-R     1451    (22)
DSX$MAPDBGBLOCK                  000006D5-R     1412    (21)
DSX$MMOFF                        00000562-R     1095    (14)
DSX$MMOFF_X                      00000578-R     1103    (14)    #-1099    (14)
DSX$MMON                         00000554-R     1045    (13)
DSX$MMON_X                       00000561-R     1051    (13)
DSX$PRINT                        00000000-XR                      202     (1)       343     (1)
DS_ERRSUP                        00000000-XR                      202     (1)       973     (12)
ENV$M_DOMAIN                    =00000002      177     (1)
ENV$M_LEVEL                     =00000001      177     (1)
ENV$M_SUPER                     =000003FC      177     (1)
ENV$S_DOMAIN                    =00000001      177     (1)
ENV$S_LEVEL                     =00000001      177     (1)
ENV$S_SUPER                     =00000008      177     (1)
ENV$V_DOMAIN                    =00000001      177     (1)       177     (1)
ENV$V_LEVEL                     =00000000      177     (1)       177     (1)
ENV$V_SUPER                     =00000002      177     (1)
ENV$_CPU                        =00000000      177     (1)       177     (1)
ENV$_FUNCTIONAL                 =00000C00      177     (1)       177     (1)
ENV$_REPAIR                     =00000001      177     (1)       177     (1)
ENV$_SUPER                      =00000001      177     (1)
ENV$_SYSTEM                     =00000001      177     (1)       177     (1)
ESTKPTR                          00000000-XR                      204     (1)     #-615     (4)
EXE$CNTREG                       00000467-R     924     (11)
EXE$CNTREG_X                     00000553-R     1002    (12)    #-997     (12)
EXE$EXPREG                       00000363-R     782     (8)
EXE$EXPREG_X                     00000466-R     864     (9)     #-793     (8)     #-811     (8)     #-855     (9)
EXE$SETPRT                       0000059D-R     1188    (17)
EXPREG$_ACMODE                  =0000000C      166     (1)
EXPREG$_NARGS                   =00000004      186     (1)
EXPREG$_PAGCNT                  =00000004      186     (1)     #-790     (8)
EXPREG$_REGION                  =00000010      186     (1)     #-804     (8)     #-834     (9)     #-857     (9)
EXPREG$_RETADR                  =00000008      186     (1)     #-854     (9)     #-863     (9)
EXPREG_COMMON                    00000373-R     787     (8)     #-780     (8)
GETBUF$_NARGS                   =00000004      178     (1)
GETBUF$_PAGCNT                  =00000004      178     (1)
GETBUF$_PHYADR                  =0000000C      178     (1)
GETBUF$_REGION                  =00000010      178     (1)
GETBUF$_RETADR                  =00000008      178     (1)
GETMEM$_NARGS                   =00000003      179     (1)
GETMEM$_PAGCNT                  =00000004      179     (1)
GETMEM$_PHYADR                  =00000008      179     (1)
GETMEM$_VRTADR                  =0000000C      179     (1)
INI$RDONLY                       00000800-R     1579    (25)
```

```
INI$WRITABLE                    00000800-R  1580  (25)
IO$GQ_PHYSICAL                  00000000-XR                  203  (1)      405  (1)    #-420  (1)
ISTKPTR                         00000000-XR                  204  (1)  #-612  (4)
KSTKPTR                         00000000-XR                  204  (1)
K_P1PAGES                      =0000007E    197  (1)    #-443  (1)  #-693  (6)      817  (8)      943  (11)
                                                          #-944  (11)
L$A_LASTADR                     00000214                     670  (6)  #-671  (6)
L_POLEN                         00000000-R  217  (1)      235  (1)  #-347  (1)  #-687  (6)  #-778  (8)
                                                          #-785  (8)  #-936  (11)
MAP$IOSPACE                     00000000-XR                  199  (1)    449  (1)
MAPFREE                         000002AA-R  664  (6)      455  (1)
MAPMEM                          00000000-R  296  (1)
MAPMEM$IOSPACE                  000001AF-R  490  (2)    #-445  (1)  #-516  (2)
MAP_BLK                         0000078D-R  1524 (25)      1424 (21)    1483 (23)
MAP_DEBUGGER                    0000074D-R  1479 (23)
MEMPOOL$INIT                    00000000-XR                  199  (1)    452  (1)
MMG$GL_SPTBASE                  00000000-XR                  200  (1)  #-378  (1)  #-703  (6)  #-821  (8)
                                                          #-948  (11)
MMG$IOLOCK                      000006C7-R  1382  (20)
MMG$UNLOCK                      000006D1-R  1387  (20)  #-1385  (20)
POPT_BASE                       00000000-XR                  199  (1)    304  (1)  #-394  (1)    600  (4)
                                                          #-682  (6)  #-685  (6)    779  (8)    784  (8)
                                                            938  (11)    964  (12)

PFN$AB_STATE                   =00000010-R  241  (1)
PFN$AB_TYPE                    =00000000    242  (1)
PFN$AL_BAK                     =00000008-R  237  (1)
PFN$AL_PTE                     =00000004-R  236  (1)
PFN$AW_BLINK                   =00000000    240  (1)
PFN$AW_FLINK                   =00000000    239  (1)
PFN$AW_REFCNT                  =0000000C-R  238  (1)
PFN$AW_SWPVBN                  =00000000-R  235  (1)
PONE                            00000621-R  1245 (18)      1213 (17)
PR$_MAPEN                      =00000038                #-1141 (15)  #-1143 (15)
PR$_POBR                       =00000008                #-1327 (19)  #-395  (1)
PR$_POLR                       =00000009                #-1232 (18)  #-348  (1)
PR$_P1BR                       =0000000A                #-1329 (19)  #-413  (1)  #-492  (2)
PR$_P1LR                       =0000000B                #-1247 (18)  #-408  (1)
PR$_SBR                        =0000000C                #-1344 (19)  #-377  (1)  #-495  (2)
PR$_SLR                        =0000000D                #-1262 (18)  #-430  (1)
PR$_TBIA                       =00000039                #-1140 (15)
PR$_TBIS                       =0000003A                #-1298 (18)
PSL$V_IS                       =0000001A    169  (1)    #-1136 (15)  #-1190 (17)
PTE$C_NA                       =00000000                  352  (1)    382  (1)  #-442  (1)    504  (2)
                                                          #-608  (4)
PTE$C_UREW                     =68000000                  #-614  (4)
PTE$C_URKW                     =70000000                    391  (1)    501  (2)  #-611  (4)
PTE$C_URSW                     =60000000                  #-617  (4)
PTE$C_UW                       =20000000                  #-1544 (25)    311  (1)  #-620  (4)  #-836  (9)
PTE$M_IO                       =00100000    193  (1)
PTE$M_MODIFY                   =04000000                  #-968  (12)
PTE$M_OWN                      =01800000                  #-1571 (25)    311  (1)    382  (1)    391  (1)
                                                          #-415  (1)  #-442  (1)    501  (2)    504  (2)
                                                          #-681  (6)  #-684  (6)  #-970  (12)  #-978  (12)
PTE$M_PFN                      =001FFFFF                  #-963  (12)
PTE$M_PROT                     =78000000                  #-1545 (25)  #-345  (1)  #-511  (2)  #-603  (4)
                                                          #-838  (9)  #-969  (12)
PTE$M_VALID                    =80000000                  #-1571 (25)    311  (1)    382  (1)    391  (1)
```

```
                                                    501     (2)     #-511    (2)     #-681    (6)     #-684    (6)
                                                  #-978     (12)
PTE$S_OWN                =00000002                #-1543    (25)    #-705    (6)     #-835    (9)     #-959    (12)
PTE$V_IO                 =00000014      193  (1)
PTE$V_OWN                =00000017                #-1543    (25)    #-705    (6)     #-835    (9)     #-959    (12)
PTE$V_VALID              =0000001F                #-1533    (25)    #-1542   (25)    #-833    (9)
PZERO                     0000060B-R    1230 (18)   1213    (17)
RELBUF$_NARGS            =00000003      181  (1)
RELBUF$_PAGCNT           =00000004      181  (1)
RELBUF$_REGION           =0000000C      181  (1)
RELBUF$_RETADR           =00000008      181  (1)
RELMEM$_NARGS            =00000003      182  (1)
RELMEM$_PAGCNT           =00000004      182  (1)
RELMEM$_PHYADR           =00000008      182  (1)
RELMEM$_VRTADR           =00000C0C      182  (1)
RESERVD                   0000064D-R    1276 (18)   1213    (17)
RPTEADR                   00000685-R    1323 (19)  #-1287    (18)    1384    (20)    1532    (25)    1541    (25)
                                                    1570    (25)    447     (1)
SCB_UNKINT                00000000-XR                204     (1)
SEP_FUNCTIONAL           =00000002      177  (1)
SEP_REPAIR               =00000003      177  (1)
SET                       00000651-R    1285 (18)  #-1257    (18)    #-1252   (18)    #-1267   (18)
SETPRT$_ACMODE           =0000000C      194  (1)
SETPRT$_INADR            =00000004      194  (1)   #-1199    (17)    #-1220   (17)
SETPRT$_NARGS            =00000005      194  (1)
SETPRT$_PROT             =00000010      194  (1)   #-1296    (18)
SETPRT$_PRVPRT           =00000014      194  (1)   #-1288    (18)    #-1292   (18)
SETPRT$_RETADR           =00000008      194  (1)   #-1194    (17)    #-1207   (17)    #-1218   (17)    #-1221   (17)
                                                  #-1299    (18)
SETPRT_XIT                0000060A-R    1222 (17)  #-1205    (17)    #-1215   (17)    #-1219   (17)
SIZ...                   =00000001      193  (1)     170    (1)       177    (1)       193    (1)
SS$_ACCVIO               =0000000C                #-1277    (18)
SS$_ILLPAGCNT            =000000FC                #-792     (8)
SS$_LENVIO               =0000018C                #-1235    (18)    #-1250   (18)    #-1265   (18)
SS$_NORMAL               =00000001                #-1303    (18)    #-1388   (20)    #-554    (3)     #-800    (8)
                                                  #-926     (11)
SS$_PAGOWNVIO            =000001EC                #-994     (12)
SS$_RESIGNAL             =00000918                #-566     (3)
SS$_VASFULL              =00000244                #-795     (8)     #-849    (9)
SSTKPTR                   00000000-XR                204     (1)     #-618    (4)
STACKPROT                 00000242-R    599  (4)     454    (1)       714    (6)
STACK_BASE                00000000-XR                204     (1)     #-601    (4)     #-606    (4)
SYMTAB_HI                 00000000-XR              #-1456    (22)    #-1458   (22)      207    (1)
SYS$UNWIND                00000000-XR                201     (1)       563    (3)
SYSTEM                    00000637-R    1260 (18)   1213    (17)
T_INSFMEM                 00000007-R    250  (1)    343     (1)
UNMAP_BLK                 000007E5-R    1567 (25)   1457    (22)      1507   (24)
UNMAP_DEBUGGER            00000766-R    1501 (24)
USTKPTR                   00000000-XR                204     (1)     #-604    (4)     #-621    (4)
XDS$INIT                 =00000000      243  (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CNTREGDEF	1	172 (1)	172 (1)
$D1_PRINT_S	1	343 (1)	343 (1)
$DEF	1	184 (1)	
$DEFINI	1	174 (1)	174 (1) 175 (1) 180 (1) 183 (1) 187 (1) 188 (1) 189 (1) 190 (1) 191 (1) 192 (1) 195 (1)
$DS_BITDEF	2	173 (1)	173 (1)
$DS_CFDEF	1	174 (1)	174 (1)
$DS_DSADEF	5	175 (1)	175 (1)
$DS_DSDEF	2	176 (1)	176 (1)
$DS_ENVDEF	2	177 (1)	177 (1)
$DS_GETBUF_DEF	1	178 (1)	178 (1)
$DS_GETMEM_DEF	1	179 (1)	179 (1)
$DS_HDRDEF	2	180 (1)	180 (1)
$DS_RELBUF_DEF	1	181 (1)	181 (1)
$DS_RELMEM_DEF	1	182 (1)	182 (1)
$DS_SCBDEF	3	183 (1)	183 (1)
$DS_TYPEDEF	4	184 (1)	184 (1)
$EQU	1	184 (1)	169 (1) 173 (1) 176 (1) 177 (1) 184 (1)
$EQULS1	1	184 (1)	169 (1) 176 (1) 177 (1) 184 (1)
$EQULST	1	169 (1)	169 (1) 176 (1) 177 (1) 184 (1)
$EXPREGDEF	1	186 (1)	186 (1)
$GBLINI	2	169 (1)	169 (1) 170 (1) 173 (1) 176 (1) 177 (1) 184 (1)
$IRPDEF	4	187 (1)	187 (1)
$OFFDEF	1	172 (1)	172 (1) 178 (1) 179 (1) 181 (1) 182 (1) 186 (1) 194 (1)
$PHDDEF	6	188 (1)	188 (1)
$PRDEF	4	189 (1)	189 (1)
$PRINT	2	340 (1)	340 (1)
$PRTDEF	1	190 (1)	190 (1)
$PSLDEF	2	191 (1)	191 (1)
$PTEDEF	3	192 (1)	192 (1)
$PUSHADR	1	973 (12)	973 (12)
$SETPRTDEF	1	194 (1)	194 (1)
$SSDEF	21	195 (1)	195 (1)
$VIELD	1	170 (1)	170 (1) 177 (1) 193 (1)
$VIELD1	1	184 (1)	170 (1) 177 (1) 193 (1)
CASE	1	1212 (17)	1212 (17)
CMK	1	1136 (15)	1136 (15) 1190 (17)
CMKDEF	1	169 (1)	169 (1)
DSFDEF	3	170 (1)	170 (1)
ERRSUP_S	1	973 (12)	973 (12)
IFNORD	1	670 (6)	670 (6)
MODNAM	1	248 (1)	248 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.13	00:00:00.29
Command processing	137	00:00:00.79	00:00:01.73
Pass 1	1202	00:00:21.83	00:00:40.26
Symbol table sort	8	00:00:02.06	00:00:03.20
Pass 2	492	00:00:05.01	00:00:10.36
Symbol table output	59	00:00:00.40	00:00:01.01
Psect synopsis output	7	00:00:00.03	00:00:00.09
Cross-reference output	115	00:00:01.62	00:00:02.33
Assembler run totals	2060	00:00:31.89	00:00:59.30

The working set limit was 1000 pages.
116759 bytes (229 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1240 non-local and 84 local symbols.
1583 source lines were read in Pass 1, producing 0 object records in Pass 2.
140 pages of virtual memory were used to define 35 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	14
DRB1:[DS.WORK]DS.MLB;218	6
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	4
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	38

1369 GETS were required to define 38 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) MEMMGT/UPDA=(MEMMGT.UPD,MEMMGT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(2) 44 DECLARATIONS
(3) 100 TU81 load device initialization
(4) 222 TU81 load driver QIO

```
                        0000     .1          .TITLE  MUBTDRIVR - TU81 LOAD DRIVER
                        0000     .2          .IDENT  '06-9'
-2                      0000      3          .DISABLE GBL
                        0000      4  ;
                        0000      5  ;*****************************************************************************
                        0000      6  ;*                                                                           *
                        0000     .1  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983, 1984 BY                            *
-1                      0000      8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
                        0000      9  ;*  ALL RIGHTS RESERVED.                                                     *
                        0000     10  ;*                                                                           *
                        0000     11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                        0000     12  ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE            *
                        0000     13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER       *
                        0000     14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
                        0000     15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
                        0000     16  ;*  TRANSFERRED.                                                             *
                        0000     17  ;*                                                                           *
                        0000     18  ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE      *
                        0000     19  ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT          *
                        0000     20  ;*  CORPORATION.                                                             *
                        0000     21  ;*                                                                           *
                        0000     22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                        0000     23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                        0000     24  ;*                                                                           *
                        0000     25  ;*                                                                           *
                        0000     26  ;*****************************************************************************
                        0000     27  ;
                        0000     28
                        0000     29  ;++
                        0000     30  ; FACILITY:     VDS
                        0000     31  ;
                        0000     32  ; ABSTRACT:
                        0000     33  ;       This module contains the load driver for the
                        0000     34  ;       TU81 magtape.
                        0000     35  ;
                        0000     36  ; ENVIRONMENT: kernel mode
                        0000     37  ;
                        0000     38  ; AUTHOR:  Marion Baggett
                        0000     39  ;
                        0000     40  ; MODIFIED BY:
                        0000     41  ;
                        0000     .1  ;
                        0000     .2  ;       01      M. Baggett      VDS 6.10
                        0000     .3  ;               Change error reporting for command execution failure.
                        0000     .4  ;
                        0000     .5  ;       02      M. Baggett      VDS 6.10
                        0000     .6  ;               Change ID field in message buffer to tape value.
                        0000     .7  ;
                        0000     .8  ;       03      M. Baggett      VDS 6.11
                        0000     .9  ;               Interchanged the offsets names in the skip record and skip file
                        0000    .10  ;               sections for marks to skip.
                        0000    .11  ;
                        0000    .12  ;       04      M. Baggett      VDS 6.11 Jan-2[illegible] 1983
                        0000    .13  ;               Change the skip file section to return a normal code on
                        0000    .14  ;               detection of tape marks.
                        0000    .15  ;
                        0000    .16  ;       05      M. Baggett      VDS 6.11 Feb-23-1983
```

```
                                0000    .17 ;
                                0000    .18 ;
                                0000    .19 ;       06      M. Baggett      VDS 6.11 June-29-1983
                                0000    .20 ;               Added GET UNIT STATUS to supply values for ONLINE command.
                                0000    .21 ;
                                0000    .22 ;       07      BOB BERGAZZI    6.12    July 18, 1983
                                0000    .23 ;               Changed b^VALID to W^VALID to resolve truncation errors.
                                0000    .24 ;
                                0000    .25 ;       08      M. Baggett      VDS 6.13 Oct 24, 1983
                                0000    .26 ;               Correct displacement error in ONLINE command packet.
                                0000    .27 ;               Set CLEAR SERIOUS EXCEPTION bit in several commands.
                                0000    .28 ;
                                0000    .29 ;       09      M. BAGGETT      VDS 6.14 Jan 5, 1984
                                0000    .30 ;               Definition of the internal processors registers are temporarily
                                C000    .31 ;               defined in this module.
                                0000     42 ;--
```

```
                                0000     44          .SBTTL  DECLARATIONS
                                0000     45 ;
                                0000     46 ; INCLUDE FILES:
                                0000     47 ;
                                0000     48
                                0000     49          $BTDDEF                         ; Boot device types
                                0000     50          $IODEF                          ; I/O function codes
                                0000     51          $MSCPDEF                        ; MSCP definitions
-1                              0000     53          $PTEDEF                         ; Page table entries
                                0000     54          $RPBDFF                         ; RPB offsets
                                0000     55          $SSDEF                          ; Status codes
                                0000     56          $UBADEF                         ; UBA definitions
                                0000     57          $UBIDEF                         ; 11/750 UBA definitions
                                0000     58          $VADEF                          ; Virtual addresses
                                C000     59
                                0000     60 ;
                                0000     61 ; EQUATED SYMBOLS:
                                0000     62 ;
                    000000B1    0000     63          BTD$K_TM        = 177   ; TEMP
                    000000B2    0000     64          BTD$K_TS        = 178   ; TEMP
                    000000B3    0000     65          BTD$K_TF        = 179   ; TEMP
                    000000B4    0000     .1          BTD$K_MU        = 180   ; TEMP                  [05]
-1                              0000     67
                    00000000    0000     68          TUIP    = 0
                    00000002    0000     69          TUSA    = 2
                    00000001    0000     70          GO      = 1
                    00008000    0000     71          OWN     = 1@15
                    0000000B    0000     72          S1      = 11
                    0000000E    0000     73          S4      = 14
                    000001EE    0000     74          UMR     = 494               ; Last-1 UNIBUS Mapping Register
                                0000     .1          .MACRO  $PRDEF,$GBL
                                0000     .2
                                0000     .3          $DEFINI PR,$GBL
                                0000     .4
                                0000     .5
                                0000     .6  $EQU    PR$_KSP 0
                                0000     .7  $EQU    PR$_ESP 1
                                0000     .8  $EQU    PR$_SSP 2
                                0000     .9  $EQU    PR$_USP 3
                                0000    .10  $EQU    PR$_ISP 4
                                0000    .11  $EQU    PR$_P0BR        8
                                0000    .12  $EQU    PR$_P0LR        9
                                0000    .13  $EQU    PR$_P1BR        10
                                0000    .14  $EQU    PR$_P1LR        11
                                0000    .15  $EQU    PR$_SBR 12
                                0000    .16  $EQU    PR$_SLR 13
                                0000    .17  $EQU    PR$_PCBB        16
                                0000    .18  $EQU    PR$_SCBB        17
                                0000    .19  $EQU    PR$_IPL 18
                                0000    .20  $EQU    PR$_ASTLVL      19
                                0000    .21  $EQU    PR$_SIRR        20
                                0000    .22  $EQU    PR$_SISR        21
                                0000    .23  $EQU    PR$_ICCS        24
                                0000    .24  $EQU    PR$_NICR        25
                                0000    .25  $EQU    PR$_ICR 26
                                0C00    .26  $EQU    PR$_TODR        27
                                0000    .27  $EQU    PR$_RXCS        32
```

```
0000    .28  $EQU   PR$_RXDB          33
0000    .29  $EQU   PR$_TXCS          34
0000    .30  $EQU   PR$_TXDB          35
0000    .31  $EQU   PR$_ACCS          40
0000    .32  $EQU   PR$_ACCR          41
0000    .33  $EQU   PR$_MAPEN         56
0000    .34  $EQU   PR$_TBIA          57
0000    .35  $EQU   PR$_TBIS          58
0000    .36  $EQU   PR$_PME 61
0000    .37  $EQU   PR$_SID 62
0000    .38  $EQU   PR$_TBCHK         63
0000    .39  $EQU   PR$V_SID_SN       0
0000    .40  $EQU   PR$S_SID_SN       12
0000    .41  $EQU   PR$V_SID_PL       12
C000    .42  $EQU   PR$S_SID_PL       3
0000    .43  $EQU   PR$V_SID_ECO      15
0000    .44  $EQU   PR$S_SID_ECO      9
0000    .45  $EQU   PR$V_SID_TYPE     24
0000    .46  $EQU   PR$S_SID_TYPE     8
0000    .47  $EQU   PR$_SID_TYP780    1
0000    .48  $EQU   PR$_SID_TYP750    2
0000    .49  $EQU   PR$_SID_TYP730    3
0000    .50  $EQU   PR$_SID_TYP7VV    4
0000    .51  $EQU   PR$_SID_TYPMAX    4
0000    .52  $EQU   PR$_WCSA          44
0000    .53  $EQU   PR$_WCSD          45
0000    .54  $EQU   PR$_SBIFS         48
0000    .55  $EQU   PR$_SBIS          49
0000    .56  $EQU   PR$_SBISC         50
0000    .57  $EQU   PR$_SBIMT         51
0000    .58  $EQU   PR$_SBIER         52
0000    .59  $EQU   PR$_SBITA         53
0000    .60  $EQU   PR$_SBIQC         54
0000    .61  $EQU   PR$_CMIERR        23
0000    .62  $EQU   PR$_CSRS          28
0000    .63  $EQU   PR$_CSRD          29
0000    .64  $EQU   PR$_CSTS          30
0000    .65  $EQU   PR$_CSTD          31
0000    .66  $EQU   PR$_TBDR          36
0000    .67  $EQU   PR$_CADR          37
0000    .68  $EQU   PR$_MCESR         38
0000    .69  $EQU   PR$_CAER          39
0000    .70  $EQU   PR$_UBRESET       55
0000    .71  $EQU   PR$_PAMACC        64
0000    .72  $EQU   PR$_PAMLOC        65
0000    .73  $EQU   PR$_CSWP          66
0000    .74  $EQU   PR$_CRBT          67
0000    .75  $EQU   PR$_MCTL1         68
0000    .76  $EQU   PR$_MCTL2         69
0000    .77  $EQU   PR$_MGEN          70
0000    .78  $EQU   PR$_MTBER         71
0000    .79  $EQU   PR$_MEAR          72
0000    .80  $EQU   PR$_MDCR1         73
0000    .81  $EQU   PR$_MEDR          74
0000    .82  $EQU   PR$_MECCR         75
0000    .83
0000    .84         $DEFEND PR,$GBL,DEF
```

```
                     0000    .85
                     0000    .86          .ENDM   $PRDEF
                     0000    .87          $PRDEF                          ; Processor registers
                     0000     75
                     0000     76 ;
                     0000     77 ; OWN STORAGE:
                     0000     78 ;
                     0000     79
                     0000     80 ;
                     0000     81 ; Boot driver table entry
                     0000     82 ;
                     0000     83
                  00000000    84 .Psect  Bootdrivr_2,    page    ; Define psect allocation
                     0000     85
                     C000     .1          $BOOT_DRIVER    DEVTYPE = BTD$K_MU,-    ; Device type (TU81)          [05]
-1                   0000     87                          SIZE = TU_DRVSIZ,-      ; Driver size
                     0000     88                          ADDR = START,-          ; Driver starting address
                     0000     89                          ENTRY = TU_DRIVER       ; Driver entry point
                     0000     90 ;                        UNIT_INIT = TU_INIT,-   ; Driver unit init entry
                     0000     .1 ;                        DRIVRNAME = TUPDRVNAME,-; Driver tape name
-1                   0000     92 ;                        AUXDRNAME = PRTDRVNAME  ; Driver port name
                     0000     93
                     0000     94 START:
                     0000     .1 ;TUPDRVNAME:
-1                   0000     96 ;        .ASCIC  /TUDRIVER.EXE/          ; Tape class driver filename
                     0000     97 ;PRTDRVNAME:
                     0000     98 ;        .ASCIC  /PTDRIVER.EXE/          ; Port driver filename
```

```
                                0000   100          .SBTTL  TU81 load device initialization
                                0000   101
                                0000   102 ;++
                                0000   103 ;
                                0000   104 ; Inputs:
                                0000   105 ;
                                0000   106 ;       R9 -->  RPB
                                0000   107 ;
                                0000   108 ; Outputs:
                                0000   109 ;
                                0000   110 ;       R0 - status code
                                0000   111 ;
                                0000   112 ;--
                                0000   113
                        01FC    C000   114 .Entry TU_INIT, ^M<R2,R3,R4,R5,R6,R7,R8>
                                0002   115
                                0002   116          .ENABLE LSB
                                0002   117
                50   38   DB  0002   118          MFPR    #PR$_MAPEN, R0          ; Get the mapping status
                                0005   119 ;
                                0005   120 ; Set up a UNIBUS mapping register(s) to cover the ring and buffers.
                                0005   121 ; To make things easy, we will grab the last two register (494 & 495).
                                0005   122 ; These registers are necessary since the ring area will be accessed by
                                0005   123 ; the controller which is a UNIBUS device that does not do any mapping.
                                0005   124 ;
          0003DC00 8F   D0      0005   125          MOVL    #<UMR@9>, R6            ; Set up a constant
                       56       000B
        52   0200'CF   9E      000C   126          MOVAB   W^INTTBL, R2            ; Get the address of the ring
          52   82'AF   CB      0011   127          BICL3   B^BYTE_OFF, R2, R1      ; Get the byte offset in page
                       51       0015
             51   56   C9      0016   128          BISL3   R6, R1, 2(R2)           ; Set in the UNIBUS addr
                  02 A2         0019
          02 A2    10'  C0      001B   129          ADDL    S^#<RING-INTTBL>,2(R2)  ; Make it the ring address
        52   15   09   EF      001F   130          EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2 ; Get the page frame
                       52       0023
                                0024   131          ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
        53   5C A940   D0      0024   132          MOVL    RPB$L_ADPPHY(R9)[R0],R3 ; Get correct pointer to TU81 reg
                                0029   133          ASSUME  RPB$L_CSRVIR EQ RPB$L_CSRPHY+4
        57   54 A940   D0      0029   134          MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get correct address of device CSR
                  0C 50  E9     002E   135          BLBC    R0,20$                  ; If clr, then physical
        52   50 B942   D0      0031   136          MOVL    @RPB$L_SVASPT(R9)[R2],R2; Virtual, get physical
         FFE00000 8F   CA      0036   137          BICL    #^C<PTE$M_PFN>,R2       ; Now a physical PFN
                       52       003C
        54   0FB8 C3   DE      003D   138 20$:     MOVAL   UBA$L_MAP+<UMR*4>(R3),R4; Get the last two UMR's
        52   00E8'CF   C9      0042    .1          BISL3   W^VALID,R2,(R4)+        ; Set as valid w/PFN                    [07]
                       84       0047
                       52   D6  0048    .2          INCL    R2                      ; Set next page just in case
        52   00E8'CF   C9      004A    .3          BISL3   W^VALID,R2,(R4)         ; Set as valid w/PFN                    [07]
                       64       004F
-3                              0050   142 ;
                                0050   143 ; Now go thru the ridiculously complicated startup sequence.  This is a
                                0050   144 ; fugue in four parts.
                                0050   145 ;
             58   02   D0      0050   146          MOVL    #2,R8                   ; Make two tries at this
        53   0200'CF   9E      0053   147 RETRY:   MOVAB   W^INTTBL, R3
             50   08   D0      0058   148          MOVL    #S1, R0                 ; Step flag
                       67   B4  005B   149          CLRW    TUIP(R7)                ; Poke the controller's CSR
```

```
                                005D   150 ;
                                005D   151 ; Wait 100 microseconds.
                                005D   152 ;
            52    1B    DB      005D   153 TIME:   MFPR    #PR$_TODR, R2           ; Current time
      52    03E8  C2    9E      0060   154         MOVAB   1000(R2), R2            ; Set for 10 seconds later
        54    02    A7    B0    0065   155 LOOP:   MOVW    TUSA(R7),R4             ; Check the status register
                    0E    19    0069   156         BLSS    ERROR                   ; Bit 15 set is the error indicator
          17  54    50    E0    006B   157         BBS     R0,R4,30$               ; Done with this step?
              51    1B    DB    006F   158         MFPR    #PR$_TODR, R1           ; No, pick up time
              52    51    D1    0072   159         CMPL    R1,R2                   ; Are we past due time?
                    02    1A    0075   160         BGTRU   ERROR                   ; Yep, error
                    EC    11    0077   161         BRB     LOOP                    ; No, try again
                D7  58    F5    0079    .1 ERROR:  SOBGTR  R8,RETRY                ; Try once again
      50    0054  8F    3C    007C    .2         MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                    04    C081    .3         RET
-1                              0082   163 BYTE_OFF:
                FFFFFE00        0082   164         .LONG   ^C<^X1FF>               ; Mask for byte offset in page
                                0086   165
        02  A7    83    B0    0086   166 30$:     MOVW    (R3)+,TUSA(R7)          ; Send the controller the next step
        CF  50    0E    F3    008A   167         AOBLEQ  #S4,R0,TIME             ; Set for next step
                                008E   168 ;
                                008E   169 ; Initialization complete.  Write the packet address in the ring.
                                008E   170 ;
      51    0250'CF   9E      008E   171         MOVAB   W^RSPPKT, R1            ; Get the address of response packet
        51    EC  AF    CA    0093   172         BICL    BYTE_OFF, R1
              51    56    C9  0097   173         BISL3   R6, R1, W^RD            ; Set byte offset in ring desc
              0210'CF         009A
      51      021C'CF   9E    009D   174         MOVAB   W^CMDPKT, R1            ; Get the address of command packet
              55    51    D0  00A2   175         MOVL    R1,R5                   ; Save pointer
        51      DA  AF    CA  00A5   176         BICL    BYTE_OFF, R1
              51    56    C9  00A9   177         BISL3   R6, R1,W^CD             ; Set byte offset in ring desc
              0214'CF         00AC
                              00AF    .1
                              00AF    .2 ;
                              00AF    .3 ; Get unit characteristics
                              00AF    .4 ;
              85    01    D0  00AF    .5         MOVL    #1,(R5)+                ; Set command ref number                [06]
        85      64  A9    9A  00B2    .6         MOVZBL  RPB$W_UNIT(R9),(R5)+    ; Put unit number in cmd packet field   [06]
              65    03    9A  00B6    .7         MOVZBL  #MSCP$K_OP_GTUNT,(R5)   ; Set opcode to get unit status         [06]
                 0285   30  00B9    .8         BSBW    EXECUTE                 ; Send it out                           [06]
                              00BC    .9
                              00BC   .10
                              00BC   178 ;
                              00BC   179 ; Now bring the device on-line
                              00BC   180 ;
      55      021C'CF   9E    00BC    .1         MOVAB   W^CMDPKT, R5            ; Get the address of command packet     [06]
              85    01    D0  00C1    .2         MOVL    #1,(R5)+                ; Set command ref number
        85      64  A9    9A  00C4    .3         MOVZBL  RPB$W_UNIT(R9),(R5)+    ; Put unit number in cmd packet field
    20000009  8F    D0  00C8    .4         MOVL    #<MSCP$M_MD_CLSEX@16>!MSCP$K_OP_ONLIN,(R5)+ ; opcode online    [08]
                    85  00CE
    0000025E'EF     9B  00CF    .5         MOVZBW  RSPPKT+MSCP$W_UNT_FLGS,2(R5)    ;                               [08]
              02    A5  00D5
                    85  D5  00D7    .6         TSTL    (R5)+                   ; Increment register                    [06]
                              00D9    .7                                         ; Set characteristics                   [06]
                    85  7C  00D9    .8         CLRQ    (R5)+                   ; Clear reserved area
                    85  D4  00DB    .9         CLRL    (R5)+                   ; Set default device dependent parameter[06]
    00000270'EF     D0  00DD   .10         MOVL    RSPPKT+MSCP$W_FORMAT,-
```

```
                      65      00E3   .11                                  (R5)    ; Set format and speed fields          [06]
-5                 025A   30  00E4   186          BSBW    EXECUTE                 ; Send it out
                       04     00E7   187          RET
                              00E8   188
-3                            00E8   192
                              00E8   193          .DISABLE LSB
                   000000E8   00E8   194          .=<.+1>&-2
                   80000000   00E8   195 VALID:   .LONG   ^X80000000             ; SIgn bit set
                              00EC   196 ;
                              00EC   197 ; RINGS
                              00EC   198 ;
                              00EC   199 ;
-1                            00EC   201 .Align  Page
                              0200   202
                       8000   C200   203 INTTBL: .WORD    OWN                   ; Step 1 pattern
                   00000000   0202   204          .LONG   0                     ; Step 2 & 3 pattern
                       0001   0206   205          .WORD   GO
                              0208   206
                  0000 0000   0208   207          .WORD   0,0                   ; Reserved
                       0000   020C   208 CMDINT:  .WORD   0                     ; Command status word
                       0000   020E   209 RSPINT:  .WORD   0                     ; Response status word
                              0210   210 RING:
                   00000000   0210   211 RD:      .LONG   0                     ; UNIBUS address of response ring
                   00000000   0214   212 CD:      .LONG   0                     ; UNIBUS address of command ring
                              0218   213 ;
                       0030   0218   214          .WORD   48                    ; Length of message
                       0101   021A    .1          .WORD   ^X101                 ; Tape ID = 1                          [02]
-1                     0001   021C   216 CMDPKT:  .WORD   1
                   0000024C   021E   217          .BLKW   23                    ; Full envelope
                              024C   218 ;
                   00000250   024C   219          .BLKW   2
                   00000280   0250   220 RSPPKT:  .BLKW   24
```

```
                     0280   222          .SBTTL  TU81 load driver QIO
                     0280   223
                     0280   224 ;++
                     0280   225 ;
                     0280   226 ; Inputs:
                     0280   227 ;
                     0280   228 ;       R3      - base address of adapter's register space
                     0280   229 ;       R5      - lbn for current piece of transfer
                     0280   230 ;       R6      - contains 0
                     0280   231 ;       R7      - address of the device's CSR
                     0280   232 ;       R8      - size of transfer in bytes
                     0280   233 ;       R9      - address of the RPB
                     0280   234 ;       R10     - starting address of transfer (byte offset in first
                     0280   235 ;                 page ORed with starting map register number)
                     0280   236 ;       R11     - LBN at start of transfer
                     0280   237 ;
                     0280   238 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                     0280   239 ;       SIZE(AP)- Size of transfer in bytes
                     0280   240 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                     0280   241 ;       BUF(AP) - Address of buffer
                     0280   242 ;
                     0280   243 ; Implicit inputs:
                     0280   244 ;
                     0280   245 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                     0280   246 ;
                     0280   247 ; Outputs:
                     0280   248 ;
                     0280   249 ;       R0 - status code
                     0280   250 ;
                     0280   251 ;               SS$_NORMAL      - successful transfer
                     0280   252 ;               SS$_CTRLERR     - fatal controller error
                     0280   253 ;               SS$_TAPEPOSLOST - tape position lost
                     0280   254 ;               SS$_DATAOVERUN  - record too long
                     0280   255 ;               SS$_ILLIOFUNC   - illegal I/O function
                     0280   256 ;               SS$_CTRLERR     - controller error
                     0280   257 ;               SS$_ENDOFFILE   - tape mark was read
                     0280   258 ;               SS$_DEVOFFLINE  - device is offline
                     0280   259 ;
                     0280   260 ;       R3 - must be preserved
                     0280   261 ;
                     0280   262 ;
                     0280   263 ;--
00000004             0280   264 BUF   = 4
00000008             0280   265 SIZE  = 8
0000000C             0280   266 LBN   = 12
00000010             0280   267 FUNC  = 16
00000014             0280   268 MODE  = 20
                     0280   269
                     0280   270
                     0280   271 ; TEMPORARY ASSIGNMENT VALUE FOR TAPE CONTROLLER(NOT IN LIBRARY YET).
                     0280   272
00000025             0280   273 MSCP$K_OP_REPOS =        37
00000001             0280   274 MSCP$M_MD_RWND  =        1
00000010             0280   275 MSCP$Z_BUFFER   =        16
00000010             0280   276 MSCP$K_ST_RDTRN =        16
00000003             0280   277 MSCP$M_MD_REVRS =        3
00000080             0280    .1 MSCP$M_MD_DLEOT =        ^X80
```

```
                    0000000C  0280     .2 MSCP$L_REC_CNT  =        12
                    00000010  0280     .3 MSCP$L_TMGP_CNT =        16
                    0000000E  0280     .4 MSCP$K_ST_TAPEM =        14
                    0000000D  0280     .5 MSCP$K_ST_BOT   =        13
                    00000011  0280     .6 MSCP$K_ST_PLOST =        17
                    00000012  0280     .7 MSCP$K_ST_LED   =        18
                    00000020  0280     .8 MSCP$W_FORMAT   =        32
-5                            0280   283
                              0280   284
                              0280   285 TU_DRIVER:                                ; TU81 device driver.
                              0280   286
                              0280    .1
-5                            0280   292 ; Translate the I/O function code into a device-dependent function
                              0280   293 ; code for this tape.
                              C280   294 ;
                 04 AE    D4  0280    .1         CLRL    4(SP)                      ; Clear retries count ********          [06]
          21     10 AC    D1  0283    .2         CmpL    Func(AP), #IO$_ReadLBlk    ; Read logical block?
-2           3B    13  0287   297         BEql    E_Read                     ; Execute read function
          24     10 AC    D1  0289   298         CmpL    Func(AP), #IO$_Rewind      ; Rewind?
                   21    13  028D   299         BEql    E_Rewind                   ; Do a rewind
          25     10 AC    D1  028F   300         CmpL    Func(AP), #IO$_SkipFile    ; Skip past eof
                   52    13  0293   301         BEql    E_SkipFile                 ; Do it
          26     10 AC    D1  0295   302         CmpL    Func(AP), #IO$_SkipRecord; Skip records?
                   03    12  0299    .1         BNEQ    10$                        ; Branch to check next function         [08]
                 007A    31  029B    .2         BRW     E_SkipRecord               ; Do it                                 [08]
          04     10 AC    D1  029E    .3 10$:    CmpL    Func(AP), #IO$_DrvClr      ; Is this drive clear?
                   07    12  02A2    .4         Bneq    20$                        ; Yes, all done, exit
                              02A4    .5 ;
                              02A4    .6 ; Start out by initing the TU81
                              02A4    .7 ;
        FD57 CF    00    FB  02A4    .8         CallS   #0, TU_Init                ; Call init code
                              02A9    .9 ;
                   04    11  02A9   .10         Brb     Return                     ; Exit
            50   F4 8F    9A  02AB   .11 20$:    MovZBL  #SS$_IllIOFunc, R0         ; illegal I/O function
-5                            02AF   308 Return:
                              05  02AF   309         RSB                        ; Return to call
                              02B0   310 ;+
                              02B0   311 ; Rewind slave drive, wait for completion
                              02B0   312 ;-
                              02B0   313 E_REWIND:
                   25     90  02B0   314         MOVB    #MSCP$K_OP_REPOS,-         ; Set function reposition
              FF6F CF          02B2   315                 CMDPKT+MSCP$B_OPCODE
              2001 8F    B0  02B5    .1         MOVW    #MSCP$M_MD_CLSEX!MSCP$M_MD_RWND,- ; Set to rewind/clear serious exce
              FF6A CF          02B9    .2                CMDPKT+MSCP$W_MODIFIER  ;
              FF68 CF    7C  02BC    .3         CLRQ    CMDPKT+MSCP$L_REC_CNT      ; Clear records/tape marks to avoid errors.
-2               007E    30  02C0   318         BSBW    EXECUTE                    ; Execute function.
                         05  02C3   319         RSB                        ; Exit.
                              02C4   320 ;+
                              02C4   321 ; Read block of data
                              02C4   322 ;-
                              02C4   323 E_READ:
                   21     90  02C4   324         MOVB    #MSCP$K_OP_READ,-          ; Set function to read
              FF5B CF          02C6   325                 CMDPKT+MSCP$B_OPCODE    ;
              2000 8F    B0  02C9    .1         MOVW    #MSCP$M_MD_CLSEX,CMDPKT+MSCP$W_MODIFIER ; Clear serious except  [08]
              FF56 CF          02CD
-1      FF53 CF    58    D0  02D0   327         MOVL    R8,CMDPKT+MSCP$L_BYTE_CNT ; Set byte count.
        FF52 CF    5A    D0  02D5   328         MOVL    R10,CMDPKT+MSCP$Z_BUFFER ; Set buffer address.
```

```
                 0064   30  02DA   329          BSBW    EXECUTE                         ; Execute function.
          51  FF7B CF   D0  02DD   330          MOVL    RSPPKT+MSCP$L_BYTE_CNT,R1 ; Bytes residual byte count.
          51  58   51   C3  02E2   331          SUBL3   R1, R8, R1                      ; From desired length is transferred.
                         05  02E6    .1          RSB                                     ; EXIT
-5                           02E7   337 ;
                             02E7   338 ;
                             02E7   339 ;+
                             02E7   340 ; Skip to end of file
                             02E7   341 ; P1 is negative if backwards else forwards
                             02E7   342 ;-
                             02E7   343 E_SKIPFILE:
                     25      90  02E7   344          MOVB    #MSCP$K_OP_REPOS,-      ; Set function to reposition.
                FF38 CF          02E9   345                  CMDPKT+MSCP$B_OPCODE
                2080 8F      B0  02EC    .1          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_CLSEX,-  ; Detect LEOT and          [08]
                FF33 CF          02F0    .2                                  CMDPKT+MSCP$W_MODIFIER  ; Clear serious except  [08]
-1           50   04 AC      D0  02F3   347          MOVL    BUF(AP),R0              ; Count of MARKS to skip.
                     0A      14  02F7   348          BGTR    10$                     ; Branch if positive.
                50   50      CE  02F9   349          MNEGL   R0,R0                   ; Make count positive.
                2083 8F      B0  02FC    .1          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_REVRS!MSCP$M_MD_CLSEX,-; Detect LEOT [08]
                FF23 CF          0300    .2                  CMDPKT+MSCP$W_MODIFIER  ; Reverse/clear serious except          [08]
          FF24 CF   50      D0  0303    .3 10$:     MOVL    R0,CMDPKT+MSCP$L_TMGP_CNT ; Set MARKS to skip.                  [03]
                FF1C CF      D4  0308    .4          CLRL    CMDPKT+MSCP$L_REC_CNT   ; Clear to prevent errors.              [08]
                   0032      30  030C    .5          BSBW    EXECUTE                 ; Execute function.
                0E   51      D1  030F    .6          CMPL    R1, #MSCP$K_ST_TAPEM    ; Was tape mark detected                [04]
                     03      12  0312    .7          BNEQ    30$                     ; No. Keep status code.                 [04]
                50   01      9A  0314    .8          MOVZBL  #SS$_NORMAL, R0         ; Change to normal if EOF               [04]
                             05  0317    .9 30$:     RSB                             ; Exit.
-5                           0318   355 ;
                             0318   356 ;+
                             0318   357 ; Skip to end of record
                             0318   358 ; P1 is negative if backward else forwards
                             0318   359 ;-
                             0318   360 E_SKIPRECORD:
                     25      90  0318   361          MOVB    #MSCP$K_OP_RFPOS,-      ; Set function to reposition.
                FF07 CF          031A   362                  CMDPKT+MSCP$B_OPCODE
                2080 8F      B0  031D    .1          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_CLSEX,-  ; Detect LEOT              [08]
                FF02 CF          0321    .2                                  CMDPKT+MSCP$W_MODIFIER  ; Clear serious except  [08]
-1           50   04 AC      D0  0324   364          MOVL    BUF(AP),R0              ; Count of RECORDS to skip.
                     0A      14  0328   365          BGTR    10$                     ; Branch if positive.
                50   50      CE  032A   366          MNEGL   R0,R0                   ; Make count positive.
                2083 8F      B0  032D    .1          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_REVRS!MSCP$M_MD_CLSEX,-; Detect LEOT [08]
                FEF2 CF          0331    .2                  CMDPKT+MSCP$W_MODIFIER  ; Reverse/clear serious except          [08]
          FEEF CF   50      D0  0334    .3 10$:     MOVL    R0,CMDPKT+MSCP$L_REC_CNT ; Set RECORD to skip.                  [03]
                FEEF CF      D4  0339    .4          CLRL    CMDPKT+MSCP$L_TMGP_CNT  ; Clear to prevent errors               [08]
                   0001      30  033D    .5          BSBW    EXECUTE                 ; Execute function.
-4                           05  0340   371          RSB                             ; Exit.
                             0341   372
                             0341   373
                             0341   374 ;+
                             0341   375 ; Execute all functions.
                             0341   376 ;-
                             0341   377 EXECUTE:
             54   02 A7      B0  0341   378          MOVW    TUSA(R7),R4             ; Controller offline?
                     77      12  0345    .1          BNEQ    50$                     ; Yep, error out
                8000 8F      A8  0347    .2          BISW    #OWN, CD+2              ; Set controller ownership
                FEC8 CF          034B
                8000 8F      A8  034E    .3          BISW    #OWN, RD+2              ; Ditto
```

```
                FEBD CF          0352
                54   67    B0    0355    .4          MOVW    TUIP(R7),R4             ; Tell controller to go.
                54    02 A7    B0    0358    .5          MOVW    TUSA(R7),R4             ; Any problems?
                           60    12    035C    .6          BNEQ    50$                     ; Yes
                54    02 A7    B0    035E    .7 30$:     MovW    TUSa(R7), R4            ; Any problems?
                           5A    12    0362    .8          BNeq    50$                     ; Yes
-8              FEAA CF    B5    0364    387         TSTW    RD+2                    ; Any response back?
                     F4    19    0368    388         BLSS    30$                     ; No, spin until there is
                           036A    389
                           036A    390 ;
                           036A    391 ; Return with  status code.
                           036A    392 ;
                05   00    EF    036A    .1 40$:     EXTZV   #0, #5, RSPPKT+MSCP$W_STATUS,R1 ; Get return status.        [04]
           51   FEEA CF          036D
-1         50   0084 8F    3C    C371    394         MovZWL  #SS$_DevOffLine, R0     ; Assume offline
                03   51    D1    0376    395         CmpL    R1, #MSCP$K_ST_OFFLN    ; Off line
                     4A    13    0379    396         BEql    60$                     ;
                50   0870 8F    3C    037B    397         MovZWL  #SS$_EndOfFile, R0      ; Assume end of file
                0E   51    D1    0380    398         CmpL    R1, #MSCP$K_ST_TAPEM    ; Tape mark
                     40    13    0383    399         BEql    60$                     ;
                12   51    D1    0385    .1          CmpL    R1, #MSCP$K_ST_LED      ; LEOT Detected             [08]
                     3B    13    0388    .2          BEQL    60$                     ;                           [08]
                50   01    D0    038A    400         MovL    #SS$_Normal, R0         ; Assume OK
                00   51    D1    038D    401         CmpL    R1, #MSCP$K_ST_SUCC     ; Success
                     33    13    0390    402         BEql    60$                     ;
                04   51    D1    0392    .1          Cmpl    R1,#MSCP$K_ST_AVLBL     ; Unit available            [08]
                     2E    13    0395    .2          Beql    60$                     ;                           [08]
                0D   51    D1    0397    .3          CmpL    R1, #MSCP$K_ST_BOT      ; At BOT
                     29    13    039A    404         BEql    60$                     ;
-1                00000838 8F    D0    039C    405         MOVL    #SS$_DataoveRun, R0     ; Assume record too large
                           50    03A2
                10   51    D1    03A3    406         CmpL    R1, #MSCP$K_ST_RDTRN    ; Data truncation
                     1D    13    03A6    407         BEql    60$                     ;
                00000224 8F    D0    03A8    408         MOVL    #SS$_TAPEPOSLOST, R0    ; Assume tape position lost.
                           50    03AE
                11   51    D1    03AF    .1          CmpL    R1, #MSCP$K_ST_PLOST    ; Tape position lost        [08]
                     11    13    03B2    .2          BEql    60$                     ;                           [08]
           50   0838 8F    3C    03B4    .3          MOVZWL  #SS$_DATAOVERUN, R0     ;                           [08]
                10   51    D1    03B9    .4          CMPL    R1, #MSCP$K_ST_RDTRN    ;                           [08]
                     07    13    03BC    .5          BEql    60$                     ;                           [08]
                00000054 8F    D0    03BE    .6 50$:     MovL    #SS$_CtrlErr, R0        ; Other codes are failure
                           50    03C4
-2                           05    03C5    411 60$:    Rsb                             ; Return
                           03C6    412
                    000003C6    03C6    413 TU_DRVSIZ=.-START
                           03C6    414
                           03C6    415         .END
```

```
$TABLE                          = 00000000 R   D  03
BTD$K_MU                        = 000000B4     D
BTD$K_TF                        = 000000B3     D
BTD$K_TM                        = 000000B1     D
BTD$K_TS                        = 000000B2     D
BUF                             = 00000004     D
BYTE_OFF                          00000082 R   D  02
CD                                00000214 R   D  02
CMDINT                            0000020C R   D  02
CMDPKT                            0000021C R   D  02
ERROR                             00000079 R   D  02
EXECUTE                           00000341 R   D  02
E_READ                            000002C4 R   D  02
E_REWIND                          000002B0 R   D  02
E_SKIPFILE                        0000C2E7 R   D  02
E_SKIPRECORD                      00000318 R   D  02
FUNC                            = 00000010     D
GO                              = 00000001     D
INITBL                            00000200 R   D  02
IO$_DRVCLR                      = 00000004     D
IO$_READLBLK                    = 00000021     D
IO$_REWIND                      = 00000024     D
IO$_SKIPFILE                    = 00000025     D
IO$_SKIPRECORD                  = 00000026     D
LBN                             = 0000000C     D
LOOP                              00000065 R   D  02
MODE                            = 00000014     D
MSCP$B_OPCODE                   = 00000008     D
MSCP$K_OP_GTUNT                 = 00000003     D
MSCP$K_OP_ONLIN                 = 00000009     D
MSCP$K_OP_READ                  = 00000021     D
MSCP$K_OP_REPOS                 = 00000025     D
MSCP$K_ST_AVLBL                 = 00000004     D
MSCP$K_ST_BOT                   = 00000005     D
MSCP$K_ST_LED                   = 00000012     D
MSCP$K_ST_OFFLN                 = 00000003     D
MSCP$K_ST_PLOST                 = 00000011     D
MSCP$K_ST_RDTRN                 = 00000010     D
MSCP$K_ST_SUCC                  = 00000000     D
MSCP$K_ST_TAPEM                 = 0000000E     D
MSCP$L_BYTE_CNT                 = 0000000C     D
MSCP$L_REC_CNT                  = 0000000C     D
MSCP$L_TMGP_CNT                 = 00000010     D
MSCP$M_MD_CLSEX                 = 00002000     D
MSCP$M_MD_DLEOT                 = 00000080     D
MSCP$M_MD_REVRS                 = 00000003     D
MSCP$M_MD_RWND                  = 00000001     D
MSCP$W_FORMAT                   = 00000020     D
MSCP$W_MODIFIER                 = 0000000A     D
MSCP$W_STATUS                   = 0000000A     D
MSCP$W_UNT_FLGS                 = 0000000E     D
MSCP$Z_BUFFER                   = 00000010     D
OWN                             = 00008000     D
PR$_MAPEN                       = 00000038     D
PR$_TODR                        = 00000018     D
PTE$M_PFN                       = 001FFFFF     D
RD                                00000210 R   D  02
RETRY                             00000053 R   D  02
RETURN                            000002AF R   D  02
RING                              00000210 R   D  02
RPB$L_ADPPHY                    = 0000005C     D
RPB$L_ADPVIR                    = 00000060     D
RPB$L_CSRPHY                    = 00000054     D
RPB$L_CSRVIR                    = 00000058     D
RPB$L_SVASPT                    = 00000050     D
RPB$W_UNIT                      = 00000064     D
RSPINT                            0000020E R   D  02
RSPPKT                            00000250 R   D  02
S1                              = 0000000B     D
S4                              = 0000000E     D
SIZE                            = 00000008     D
SS$_CTRLERR                     = 00000054     D
SS$_DATAOVERUN                  = 00000838     D
SS$_DEVOFFLINE                  = 00000084     D
SS$_ENDOFFILE                   = 00000870     D
SS$_ILLIOFUNC                   = 000000F4     D
SS$_NORMAL                      = 00000001     D
SS$_TAPEPOSLOST                 = 00000224     D
START                             00000000 R   D  02
TIME                              0000005D R   D  02
TUIP                            = 00000000     D
TUSA                            = 00000002     D
TU_DRIVER                         00000280 R   D  02
TU_DRVSIZ                       = 000003C6     D
TU_INIT                           00000000 RG  D  02
UBA$L_MAP                       = 00000800     D
UMR                             = 000001EE     D
VA$S_VPN                        = 00000015     D
VA$V_VPN                        = 00000009     D
VALID                             000000E8 R   D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	000003C6 (966.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
BOOTDRIVR_4	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	89 (2)	89 (2)
BTD$K_MU	=000000B4	65.1 (2)	89 (2)
BTD$K_TF	=000000B3	65 (2)	
BTD$K_TM	=000000B1	63 (2)	
BTD$K_TS	=000000B2	64 (2)	
BUF	=00000004	264 (4)	#-347 (4) #-364 (4)
BYTE_OFF	00000082-R	163 (3)	#-127 (3) #-172 (3) #-176 (3)
CD	00000214-R	212 (3)	#-177 (3) #-378.2 (4)
CMDINT	0000020C-R	208 (3)	
CMDPKT	0000021C-R	216 (3)	174 (3) 180.1 (3) #-315 (4) #-315.2 (4) #-315.3 (4) #-325 (4) #-325.1 (4) #-327 (4) #-328 (4) #-345 (4) #-345.2 (4) #-349.2 (4) #-349.3 (4) #-349.4 (4) #-362 (4) #-362.2 (4) #-366.2 (4) #-366.3 (4) #-366.4 (4)
ERROR	00000079-R	161.1 (3)	#-156 (3) #-160 (3)
EXECUTE	00000341-R	377 (4)	#-177.8 (3) #-186 (3) #-318 (4) #-329 (4) #-349.5 (4) #-366.5 (4)
E_READ	000002C4-R	323 (4)	#-297 (4)
E_REWIND	000002B0-R	313 (4)	#-299 (4)
E_SKIPFILE	000002E7-R	343 (4)	#-301 (4)
E_SKIPRECORD	00000318-R	360 (4)	#-302.2 (4)
FUNC	=00000010	267 (4)	#-294.2 (4) #-298 (4) #-300 (4) #-302 (4) #-302.3 (4)
GO	=00000001	70 (2)	205 (3)
INTTBL	00000200-R	203 (3)	126 (3) #-129 (3) 147 (3)
IO$_DRVCLR	=00000004		#-302.3 (4)
IO$_READLBLK	=00000021		#-294.2 (4)
IO$_REWIND	=00000024		#-298 (4)
IO$_SKIPFILE	=00000025		#-300 (4)
IO$_SKIPRECORD	=00000026		#-302 (4)
LBN	=0000000C	266 (4)	
LOOP	00000065-R	155 (3)	#-161 (3)
MODE	=00000014	268 (4)	
MSCP$B_OPCODE	=00000008		#-315 (4) #-325 (4) #-345 (4) #-362 (4)
MSCP$K_OP_GTUNT	=00000003		#-177.7 (3)
MSCP$K_OP_ONLIN	=00000009		#-180.4 (3)
MSCP$K_OP_READ	=00000021		#-324 (4)
MSCP$K_OP_REPOS	=00000025	273 (4)	#-314 (4) #-344 (4) #-361 (4)
MSCP$K_ST_AVLBL	=00000004		#-402.1 (4)
MSCP$K_ST_BOT	=0000000D	277.5 (4)	#-402.3 (4)
MSCP$K_ST_LED	=00000012	277.7 (4)	#-399.1 (4)
MSCP$K_ST_OFFLN	=00000003		#-395 (4)
MSCP$K_ST_PLOST	=00000011	277.6 (4)	#-408.1 (4)
MSCP$K_ST_RDTRN	=00000010	276 (4)	#-406 (4) #-408.4 (4)
MSCP$K_ST_SUCC	=00000000		#-401 (4)
MSCP$K_ST_TAPEM	=0000000E	277.4 (4)	#-349.6 (4) #-398 (4)
MSCP$L_BYTE_CNT	=0000000C		#-327 (4) #-330 (4)
MSCP$L_REC_CNT	=0000000C	277.2 (4)	#-315.3 (4) #-349.4 (4) #-366.3 (4)
MSCP$L_TMGP_CNT	=00000010	277.3 (4)	#-349.3 (4) #-366.4 (4)
MSCP$M_MD_CESEX	=00002000		#-180.4 (3) #-315.1 (4) #-325.1 (4) #-345.1 (4)

```
                                                        #-349.1  (4)    #-362.1  (4)    #-366.1  (4)
MSCP$M_MD_DLEOT     =00000080     277.1  (4)            #-345.1  (4)    #-349.1  (4)    #-362.1  (4)    #-366.1  (4)
MSCP$M_MD_REVRS     =00000003     277    (4)            #-349.1  (4)    #-366.1  (4)
MSCP$M_MD_RWND      =00000001     274    (4)            #-315.1  (4)
MSCP$W_FORMAT       =00000020     277.8  (4)            #-180.10 (3)
MSCP$W_MODIFIER     =0000000A                           #-315.2  (4)    #-325.1  (4)    #-345.2  (4)    #-349.2  (4)
                                                        #-362.2  (4)    #-366.2  (4)
MSCP$W_STATUS       =0000000A                             392.1  (4)
MSCP$W_UNT_FLGS     =0000000E                           #-180.5  (3)
MSCP$Z_BUFFER       =00000010     275    (4)            #-328    (4)
OWN                 =00008000     71     (2)              203    (3)    #-378.2  (4)    #-378.3  (4)
PR$_MAPEN           =00000038                           #-118    (3)
PR$_TODR            =0000001B                           #-153    (3)    #-158    (3)
PTE$M_PFN           =001FFFFF                           #-137    (3)
RD                   00000210-R   211    (3)            #-173    (3)    #-378.3  (4)    #-387    (4)
RETRY                00000053-R   147    (3)            #-161.1  (3)
RETURN               000002AF-R   308    (4)            #-302.10 (4)
RING                 00000210-R   210    (3)            #-129    (3)
RPB$L_ADPPHY        =0000005C                             131    (3)    #-132    (3)
RPB$L_ADPVIR        =00000060                             131    (3)
RPB$L_CSRPHY        =00000054                             133    (3)    #-134    (3)
RPB$L_CSRVIR        =00000058                             133    (3)
RPB$L_SVASPT        =00000050                           #-136    (3)
RPB$W_UNIT          =00000064                           #-177.6  (3)    #-180.3  (3)
RSPINT               0000020E-R   209    (3)
RSPPKT               00000250-R   220    (3)              171    (3)    #-180.10 (3)    #-180.5  (3)    #-330    (4)
                                                          392.1  (4)
S1                  =0000000B     72     (2)            #-148    (3)
S4                  =0000000E     73     (2)            #-167    (3)
SIZE                =00000008     265    (4)
SS$_CTRLERR         =00000054                           #-161.2  (3)    #-408.6  (4)
SS$_DATAOVERUN      =00000838                           #-405    (4)    #-408.3  (4)
SS$_DEVOFFLINE      =00000084                           #-394    (4)
SS$_ENDOFFILE       =00000870                           #-397    (4)
SS$_ILLIOFUNC       =000000F4                           #-302.11 (4)
SS$_NORMAL          =00000001                           #-349.8  (4)    #-400    (4)
SS$_TAPEPOSLOST     =00000224                           #-408    (4)
START                00000000-R   94     (2)              413    (4)      89     (2)
TIME                 0000005D-R   153    (3)            #-167    (3)
TUIP                =00000000     68     (2)            #-149    (3)    #-378.4  (4)
TUSA                =00000002     69     (2)            #-155    (3)    #-166    (3)    #-378    (4)    #-378.5  (4)
                                                        #-378.7  (4)
TU_DRIVER            00000280-R   285    (4)              89     (2)
TU_DRVSIZ           =000003C6     413    (4)              89     (2)
TU_INIT              00000000-R   114    (3)             302.8  (4)
UBA$L_MAP           =00000800                             138    (3)
UMR                 =000001EE     74     (2)            #-125    (3)      138    (3)
VA$S_VPN            =00000015                           #-130    (3)
VA$V_VPN            =00000009                           #-130    (3)
VALID                000000E8-R   195    (3)            #-138.1  (3)    #-138.3  (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	85.1 (2)	85.1 (2)
$BTDDEF	1	49 (2)	49 (2)
$DEFINI	1	49 (2)	49 (2) 50 (2) 51 (2) 53 (2) 54 (2) 55 (2) 56 (2) 57 (2) 58 (2) 74.87 (2)
$IODEF	17	50 (2)	50 (2)
$MSCPDEF	18	51 (2)	51 (2)
$PRDEF	4	74.1 (2)	74.87 (2)
$PTEDEF	3	53 (2)	53 (2)
$RPBDEF	5	54 (2)	54 (2)
$SSDEF	21	55 (2)	55 (2)
$UBADEF	6	56 (2)	56 (2)
$UBIDEF	2	57 (2)	57 (2)
$VADEF	1	58 (2)	58 (2)
ASSUME	1	131 (3)	131 (3) 133 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.10	00:00:00.89
Command processing	148	00:00:00.84	00:00:03.23
Pass 1	606	00:00:17.97	00:00:24.44
Symbol table sort	0	00:00:02.43	00:00:03.07
Pass 2	165	00:00:04.21	00:00:07.48
Symbol table output	12	00:00:00.11	00:00:00.11
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	27	00:00:00.44	00:00:00.48
Assembler run totals	1001	00:00:26.13	00:00:39.73

The working set limit was 1000 pages.
79780 bytes (156 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1573 non-local and 11 local symbols.
563 source lines were read in Pass 1, producing 0 object records in Pass 2.
56 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	7
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	14

1577 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) MUBTDRIVR/UPDA=(MUBTDRIVR.UPD,MUBTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

```
;  0001  0     %title '*** ODS Disk RMS routines'
;  0002  0     module ODS (                                ! RMS$OPEN, RMS$GET, RMS$READ, RMS$CLOSE services
;  0003  0                 ident = '01-04'
;  0004  0                 ) =
;  0005  0     !
;  0006  0     ! Copyright (c) 1980, 1982
;  0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;  0008  0     !
;  0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;  0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;  0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;  0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;  0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;  0014  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;  0015  0     ! REMAIN IN DEC.
;  0016  0     !
;  0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;  0018  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;  0019  0     ! CORPORATION.
;  0020  0     !
;  0021  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;  0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;  0023  0     !
;  0024  0     !++
;  0025  0     ! FACILITY: Diagnostic Supervisor
;  0026  0     !
;  0027  0     ! ABSTRACT:
;  0028  0     !
;  0029  0     !       This module supports RMS functions on ODS media.
;  0030  0     !
;  0031  0     ! AUTHOR:
;  0032  0     !       Dave Butenhof,          19-aug-1980
;  0033  0     !
;  0034  0     ! MODIFIED BY:
;  0035  0     !
;  0036  0     !               Dave Butenhof, 08-oct-1980, version 6.1
;  0037  0     !       1       Specify "long_relative" access to BOO$QIO routine
;  0038  0     !               Dave Butenhof, 6-feb-1981, version 6.3
;  0039  0     !       2       Fix so full record length if returned in RAB$L_STV on
;  0040  0     !               RMS$_RTB errors.
;  0041  0     !               - Dave Butenhof, 02-Jun-1981, version 6.4
;  0042  0     !       03      Alter library spec for flexibility
;  0043  0     !
;  0044  0     !       04      - Dave Butenhof, 13-Jan-1982, version 6.6
;  0045  0     !               Make some minor fixes like variable attributes.  Also,
;  0046  0     !               support ODS-1 by taking RSIZE field of file neader if
;  0047  0     !               MAXREC is 0 (as it is for ODS-1 fixed length).
;  0048  0     !--
;  0049  0
```

```
;    0050  0     %sbttl 'declarations'
;    0051  1     begin
;    0052  1     !
;    0053  1     ! include files:
;    0054  1     !
;    0055  1
;    0056  1     library '$diag';                !                                                                   [03]
;    0057  1
;    0058  1     library '$ds';                  !                                                                   [03]
;    0059  1
;    0060  1     library 'sys$library:lib.l32';
;    0061  1
;    0062  1     !
;    0063  1     ! local declarations
;    0064  1     !
;    0065  1
;    0066  1     linkage
;    0067  1         jsb_ods = jsb (register = 0) : global (block = 6, vbn = 7, offset = 8, cblock = 9, rab = 10);
;    0068  1
;    0069  1     !
;    0070  1     ! table of contents:
;    0071  1     !
;    0072  1
;    0073  1     forward routine
;    0074  1         ods$advance : jsb_ods novalue,          ! Advance RFA
;    0075  1         ods$access : jsb_ods,                   ! Access a VBN
;    0076  1         ods$open : call_rms,                    ! RMS$OPEN emulator--open file
;    0077  1         ods$get : call_rms,                     ! RMS$GET emulator--deblock record
;    0078  1         ods$read : call_rms;                    ! RMS$READ emulator--read file block(s)
;    0079  1
;    0080  1     !
;    0081  1     ! Local definitions
;    0082  1     !
;    0083  1
;    0084  1     ! External references
;    0085  1     !
;    0086  1
;    0087  1     external routine
;    0088  1         rms$alloc_cache : jsb_cblock,       ! Allocate cache blocks
;    0089  1         fil$openfile,                       ! 'Open' a file
;    0090  1         fil$readvbn,                        ! Read a virtual block
;    0091  1         boo$qio : addressing_mode (long_relative),  ! standalone 'level-3' QIO
;    0092  1         exe$alononpaged : jsb_alloc addressing_mode (long_relative);        ! Allocate static buffer space
;    0093  1
;    0094  1     !
;    0095  1     ! psect definitions:
;    0096  1     !
;    0097  1
;    0098  1     psect
;    0099  1         plit = data ( share, addressing_mode (long_relative));
;    0100  1
;    0101  1     psect
;    0102  1         own = work ( read, write);
;    0103  1
;    0104  1     psect
;    0105  1         global = work ( read, write);
;    0106  1
```

```
;   0107  1     psect
;   0108  1         code = code (share);
;   0109  1
```

```
;    0110  1     %sbttl 'Advance RFA'
;    0111  1     routine ods$advance (len) : jsb_ods novalue =
;    0112  1
;    0113  1     !++
;    0114  1     ! Functional description:
;    0115  1     !       Increment the internal 'RFA' (offset and vbn parts) by the
;    0116  1     !       specified number of bytes, considering round-off and block
;    0117  1     !       overlap.
;    0118  1     !
;    0119  1     ! Inputs:
;    0120  1     !       len                     number of bytes to increment by
;    0121  1     !
;    0122  1     ! Implicit inputs:
;    0123  1     !       R7                      vbn
;    0124  1     !       R8                      offset in vbn
;    0125  1     !
;    0126  1     ! Outputs:
;    0127  1     !       none
;    0128  1     !
;    0129  1     ! Implicit outputs:
;    0130  1     !       R7                      new vbn
;    0131  1     !       R8                      new byte offset in vbn
;    0132  1     !
;    0133  1     ! Side effects:
;    0134  1     !       none
;    0135  1     !
;    0136  1     ! Return codes:
;    0137  1     !       none
;    0138  1     !--
;    0139  1
;    0140  2         begin
;    0141  2
;    0142  2         external register
;    0143  2             vbn = 7,                            ! Virtual block number
;    0144  2             offset = 8;                         ! Byte offset in VBN
;    0145  2
;    0146  2         local
;    0147  2             bite;
;    0148  2
;    0149  2         bite = (.offset + .len + 1) and not 1;  ! Add length and round up
;    0150  2         offset = .bite and 511;                 ! Truncate to offset in page
;    0151  3         vbn = .vbn + (.bite/512)                ! Calculate new page
;    0152  1         end;                                    ! of ods$advance

                                                            .TITLE  ODS *** ODS Disk RMS routines
                                                            .IDENT  \01-04\

                                                            .EXTRN  RMS$ALLOC_CACHE
                                                            .EXTRN  FIL$OPENFILE, FIL$READVBN
                                                            .EXTRN  BOO$QIO, EXE$ALONONPAGED

                                                            .PSECT  CODE,NOWRT,  SHR,2

                                50      01 A048  9E 00000 ODS$ADVANCE:
                                                            MOVAB   1(LEN)[OFFSET], R0                  ; 0149
                                50           01  8A 00005           BICB2   #1, BITE                   ;
```

```
                58              50              09              00  EF 00008        EXTZV   #0, #9, BITE, OFFSET                    ; 0150
                                                50 00000200     8F  C6 0000D        DIVL2   #512, R0                                ; 0151
                                                57              50  C0 00014        ADDL2   R0, VBN                                 ;
                                                                    05 00017        RSB                                             ; 0152

; Routine Size:  24 bytes,    Routine Base:  CODE + 0000
```

```
;    0153  1     %sbttl 'Access file block'
;    0154  1     routine ods$access : jsb_ods =
;    0155  1
;    0156  1     !++
;    0157  1     ! Functional description:
;    0158  1     !       If the desired VBN is not currently cached, update cache to
;    0159  1     !       contain it.
;    0160  1     !
;    0161  1     ! Inputs:
;    0162  1     !       none
;    0163  1     !
;    0164  1     ! Implicit inputs:
;    0165  1     !       R7                      VBN to cache/access
;    0166  1     !       R9                      pointer to CBLOCK control structure
;    0167  1     !       R10                     pointer to RAB
;    0168  1     !
;    0169  1     ! Outputs:
;    0170  1     !       none
;    0171  1     !
;    0172  1     ! Implicit outputs:
;    0173  1     !       R6                      pointer to 'memory image' of file block
;    0174  1     !
;    0175  1     ! Side effects:
;    0176  1     !       may alter cache control structure of CBLOCK
;    0177  1     !
;    0178  1     ! Return codes:
;    0179  1     !       RMS$_NORMAL             OK
;    0180  1     !       RMS$_RFA                random RFA is illegal
;    0181  1     !       RMS$_EOF                VBN is past end of file
;    0182  1     !       RMS$_RER                File read error
;    0183  1     !--
;    0184  1
;    0185  2         begin
;    0186  2
;    0187  2         external register
;    0188  2             block = 6,
;    0189  2             vbn = 7,
;    0190  2             cblock = 9 : ref bblock,
;    0191  2             rab = 10 : ref bblock;
;    0192  2
;    0193  2         local
;    0194  2             read_blocks;                                ! Blocks to EOF
;    0195  2
;    0196  2         bind
;    0197  2             cache = cblock [ctl$l_cache1] : vector; ! Map for convenience
;    0198  2
;    0199  2         read_blocks = .cblock [ctl$l_eofvbn] - .vbn;
;    0200  2
;    0201  2         if .read_blocks lss 0                           ! If no more blocks
;    0202  3             or (.read_blocks eql 0                      !  or at EOF block
;    0203  3             and .cblock [ctl$w_offbyte] eql 0)          !  and no bytes there
;    0204  2         then
;    0205  2             return
;    0206  2
;    0207  2                 if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof;      ! Check for illegal vbn
;    0208  2
;    0209  2         if .vbn lss .cblock [ctl$l_lvbn]                ! If VBN isn't
```

```
;    0210  2            or .vbn geq .cblock [ctl$l_hvbn]          !  cached now
;    0211  2        then
;    0212  3            begin                                     ! then cache it
;    0213  3
;    0214  3            if .cblock [ctl$w_cache_size] eql 0       ! If no cache allocated
;    0215  3            then
;    0216  4                begin
;    0217  4
;    0218  4                local
;    0219  4                    stat;
;    0220  4
;    0221  4                stat = rms$alloc_cache (1);           ! Allocate cache
;    0222  4
;    0223  4                if not .stat
;    0224  4                then
;    0225  4                    return rms$_dme                   ! Error if can't
;    0226  3                end;
;    0227  3
;    0228  3            cblock [ctl$l_lvbn] = 0;                  ! Clear start vbn
;    0229  3            cblock [ctl$l_hvbn] = 0;                  ! Clear high vbn
;    0230  3            read_blocks = min (2,                     ! 3 blocks or to EOF
;    0231  3                (.read_blocks*512 + .cblock [ctl$w_offbyte] + 511)/512 - 1);
;    0232  3
;    0233  3            incr i from 0 to .read_blocks do          ! read cache buffers
;    0234  4                begin
;    0235  4
;    0236  4                local
;    0237  4                    stat;
;    0238  4
;    0239  5                stat = (if .cblock [ctl$l_startlbn] neq 0   ! If contiguous
;    0240  5                then boo$qio (.cache [.i], 512,             ! do direct QIO
;    0241  5                        .vbn + .cblock [ctl$l_startlbn] + .i - 1, io$_readlblk, 0, .cblock [ctl$l_rpb]) else
;    0242  5                    fil$readvbn (.cblock [ctl$l_rpb], .vbn + .i,    ! Translate VBN to
;    0243  4                        .cache [.i], .cblock [ctl$l_filhdr]));      !  LBN and read
;    0244  4
;    0245  4                if not .stat                          ! If it failed
;    0246  4                then
;    0247  5                    begin
;    0248  5                    rab [rab$l_stv] = .stat;          ! Load QIO return
;    0249  5                    return rms$_rer
;    0250  5                    end
;    0251  5
;    0252  3                end;
;    0253  3
;    0254  3            cblock [ctl$l_lvbn] = .vbn;               ! Set new low vbn
;    0255  3            cblock [ctl$l_hvbn] = .vbn + .read_blocks + 1   ! Set new high vbn
;    0256  2            end;
;    0257  2
;    0258  2        block = .cache [.vbn - .cblock [ctl$l_lvbn]];       ! Get buffer pointer
;    0259  2        1                                             ! Return success
;    0260  1        end;                                          ! of ods$access
```

```
                                          PUSHR   #^M<R2,R3,R4>                          ; 0154
           54       2C   A9  9E 00002     MOVAB   44(CBLOCK), R4                         ; 0197
 53     44  A9           57  C3 00006     SUBL3   VBN, 68(CBLOCK), READ_BLOCKS           ; 0199
                         07  19 0000B     BLSS    1$                                     ; 0201
                         1D  12 0000D     BNEQ    3$                                     ; 0202
                    42   A9  B5 0000F     TSTW    66(CBLOCK)                             ; 0203
                         18  12 00012     BNEQ    3$                                     ;
           02       1E   AA  91 00014 1$: CMPB    30(RAB), #2                            ; 0207
                         09  12 00018     BNEQ    2$                                     ;
           50 0001865C   8F  D0 0001A     MOVL    #99932, R0                             ;
                         2D  11 00021     BRB     5$                                     ;
           50 0001827A   8F  D0 00023 2$: MOVL    #98938, R0                             ;
                         24  11 0002A     BRB     5$                                     ;
        28  A9           57  D1 0002C 3$: CMPL    VBN, 40(CBLOCK)                        ; 0209
                         09  19 00030     BLSS    4$                                     ;
        24  A9           57  D1 00032     CMPL    VBN, 36(CBLOCK)                        ; 0210
                         03  18 00036     BGEQ    4$                                     ;
                       009D  31 00038     BRW     13$                                    ;
                    0A   A9  B5 0003B 4$: TSTW    10(CBLOCK)                             ; 0214
                         12  12 0003E     BNEQ    6$                                     ;
           50            01  D0 00040     MOVL    #1, R0                                 ; 0221
                      0000G  30 00043     BSBW    RMS$ALLOC_CACHE                        ;
           09            50  E8 00046     BLBS    STAT, 6$                               ; 0223
           50 000184D4   8F  D0 00049     MOVL    #99540, R0                             ; 0225
                         76  11 00050 5$: BRB     11$                                    ;
                    24   A9  7C 00052 6$: CLRQ    36(CBLOCK)                             ; 0229
 50        53            09  78 00055     ASHL    #9, READ_BLOCKS, R0                    ; 0231
           51       42   A9  3C 00059     MOVZWL  66(CBLOCK), R1                         ;
           50     01FF C140  9E 0005D     MOVAB   511(R1)[R0], R0                        ;
           50 00000200   8F  C6 00063     DIVL2   #512, R0                               ;
                         50  D7 0006A     DECL    R0                                     ;
           02            50  D1 0006C     CMPL    R0, #2                                 ; 0230
                         03  15 0006F     BLEQ    7$                                     ;
           50            02  D0 00071     MOVL    #2, R0                                 ;
           53            50  D0 00074 7$: MOVL    R0, READ_BLOCKS                        ;
           52            01  CE 00077     MNEGL   #1, I                                  ; 0241
                         4E  11 0007A     BRB     12$                                    ;
 51        52            02  78 0007C 8$: ASHL    #2, I, R1                              ; 0240
                    50   A9  D5 00080     TSTL    80(CBLOCK)                             ; 0239
                         22  13 00083     BEQL    9$                                     ;
                    38   A9  DD 00085     PUSHL   56(CBLOCK)                             ; 0241
           7E            21  7D 00088     MOVQ    #33, -(SP)                             ; 0240
 50        57       50   A9  C1 0008B     ADDL3   80(CBLOCK), VBN, R0                    ; 0241
                    FF A240  9F 00090     PUSHAB  -1(I)[R0]                              ;
           7E     0200   8F  3C 00094     MOVZWL  #512, -(SP)                            ; 0240
                       6144  9F 00099     PUSHAB  (R1)[R4]                               ;
                         9E  DD 0009C     PUSHL   @(SP)+                                 ;
     00000000G  EF       06  FB 0009E     CALLS   #6, BOO$QIO                            ;
                         13  11 000A5     BRB     10$                                    ;
                    54   A9  DD 000A7 9$: PUSHL   84(CBLOCK)                             ; 0243
                       6144  9F 000AA     PUSHAB  (R1)[R4]                               ;
                         9E  DD 000AD     PUSHL   @(SP)+                                 ;
                       6247  9F 000AF     PUSHAB  (I)[VBN]                               ; 0242
                    38   A9  DD 000B2     PUSHL   56(CBLOCK)                             ;
         0000G  CF       04  FB 000B5     CALLS   #4, FIL$READVBN                        ;
           0D            50  E8 000BA 10$: BLBS   STAT, 12$                              ; 0245
       0C   AA           50  D0 000BD     MOVL    STAT, 12(RAB)                          ; 0248
```

```
                                50 0001C0F4   8F  D0 000C1          MOVL    #114932, R0                                     ; 0249
                                              1A  11 000C8 11$:      BRB     14$                                             ;
                     AE           52          53  F3 000CA 12$:     AOBLEQ  READ_BLOCKS, I, 8$                              ; 0245
                        28        A9          57  D0 000CE          MOVL    VBN, 40(CBLOCK)                                 ; 0254
                        24        A9    01 A347  9E 000D2          MOVAB   1(READ_BLOCKS)[VBN], 36(CBLOCK)                 ; 0255
                     50           57    28   A9  C3 000D8 13$:     SUBL3   40(CBLOCK), VBN, R0                             ; 0258
                                  56        6440  D0 000DD          MOVL    (R4)[R0], BLOCK                                 ;
                                  50          01  D0 000E1          MOVL    #1, R0                                          ; 0260
                                              1C  BA 000E4 14$:     POPR    #^M<R2,R3,R4>                                   ;
                                                  05 000E6          RSB                                                     ;

; Routine Size:  231 bytes,    Routine Base:  CODE + 0018
```

```
;    0261  1     %sbttl 'ODS OPEN service'
;    0262  1
;    0263  1     global routine ods$open : call_rms =
;    0264  1
;    0265  1     !++
;    0266  1     ! Functional description:
;    0267  1     !       Access a file on ODS files-11 disk media.  Return file parameters
;    0268  1     !       in the FAB and internal control block associated with the file
;    0269  1     !
;    0270  1     ! Inputs:
;    0271  1     !       none
;    0272  1     !
;    0273  1     ! Implicit inputs:
;    0274  1     !       R11     points to the FAB
;    0275  1     !       R9      points to the CBLOCK (internal RMS control block)
;    0276  1     !
;    0277  1     ! Outputs:
;    0278  1     !       none
;    0279  1     !
;    0280  1     ! Implicit outputs:
;    0281  1     !       Several fields in the FAB and in the static file control block
;    0282  1     !       are altered to describe the file.  Also, if an XAB (XABFHC is
;    0283  1     !       the only XAB currently supported) exists, it will be filled in.
;    0284  1     !
;    0285  1     ! Return status
;    0286  1     !       RMS$_DME        insufficient memory for buffer allocation
;    0287  1     !       RMS$_FNF        file not found
;    0288  1     !       RMS$_FNM        Bad file name
;    0289  1     !       RMS$_RER        QIO error on file access
;    0290  1     !       RMS$_NORMAL     success
;    0291  1     !--
;    0292  1
;    0293  2         begin
;    0294  2
;    0295  2         local
;    0296  2             ndxhdr : bblock [512],              ! Buffer for index header
;    0297  2             filhdr : ref bblock,                ! Convenient pointer
;    0298  2             status : vector [2],                ! BOO$QIO status return
;    0299  2             stat;
;    0300  2
;    0301  2         external register
;    0302  2             cblock = 9 : ref bblock,            ! Pointer to control block
;    0303  2             rab = 10,
;    0304  2             fab = 11 : ref bblock;              ! Define the FAB
;    0305  2
;    0306  3         begin                                   ! Allocate file header space
;    0307  3
;    0308  3         global register
;    0309  3             size = 1,
;    0310  3             addr = 2;
;    0311  3
;    0312  3         size = 512;                             ! Size of file header
;    0313  3
;    0314  3         if not exe$alononpaged () then return rms$_dme;
;    0315  3
;    0316  3         filhdr = cblock [ctl$l_filhdr] = .addr      ! Remember address
;    0317  2         end;
```

```
;    0318  2          stat = fil$openfile (.cblock [ctl$l_rpb], cblock [ctl$q_filename], ndxhdr, .filhdr, status);
;    0319  2                                                          ! Access the file
;    0320  2
;    0321  2          if not .stat                                    ! Error
;    0322  2          then
;    0323  3              return (selectone .stat of
;    0324  3                      set
;    0325  3                      [ss$_nosuchfile] : rms$_fnf;
;    0326  3                      [ss$_badfilename] : rms$_fnm;
;    0327  3                      [ss$_filestruct, ss$_badfilehdr, ss$_badchksum] : (fab [fab$l_stv] = .stat; rms$_acc);
;    0328  4                      [otherwise] : (fab [fab$l_stv] = .stat; rms$_rer)
;    0329  2                      tes);
;    0330  2
;    0331  2          if .filhdr [fh2$b_struclev] eql 2               ! If structure level 2                                  [04]
;    0332  2          then
;    0333  2              cblock [ctl$v_ods2] = 1;                    ! Then set flag bit
;    0334  2
;    0335  2          cblock [ctl$l_get] = ods$get;                   ! Set GET routine address
;    0336  2          cblock [ctl$l_read] = ods$read;                 ! Set READ routine address
;    0337  2          cblock [ctl$l_startlbn] = .status [0];          ! First LBN (if contiguous)
;    0338  3          begin                                           ! Load file attributes
;    0339  3
;    0340  3          builtin
;    0341  3              rot;
;    0342  3
;    0343  3          bind
;    0344  3              fileatt = (if .cblock [ctl$v_ods2] then filhdr [fh2$w_recattr] else filhdr [fh1$w_recattr]) : bblock;
;    0345  3
;    0346  3          fab [fab$b_rat] = .fileatt [fat$b_rattrib]; ! Copy record attributes
;    0347  3                                                      ! Get EOF block, which is
;    0348  3          cblock [ctl$l_eofvbn] =                     ! The EOF block VBN is
;    0349  3          rot (.fileatt [fat$l_efblk], 16);           !  stored sideways
;    0350  3          cblock [ctl$w_offbyte] = .fileatt [fat$w_ffbyte];   ! First free byte
;    0351  3          fab [fab$b_org] = .fileatt [fat$v_fileorg];  ! File organization
;    0352  3          fab [fab$b_rfm] = .fileatt [fat$v_rtype];    ! Record format
;    0353  3          fab [fab$w_mrs] = .fileatt [fat$w_maxrec];   ! Max. rec size
;    0354  3
;    0355  3          If .Fab [Fab$W_Mrs] Eql 0                   ! If no MAXREC                                          [04]
;    0356  3          Then                                                                                                [04]
;    0357  3              Fab [Fab$W_Mrs] = .FileAtt [Fat$W_RSize];       ! Get the record size                           [04]
;    0358  3
;    0359  3          fab [fab$b_fsz] = .fileatt [fat$b_vfcsize]  ! Variable/fixed header size
;    0360  2          end;
;    0361  2          rms$_normal
;    0362  1          end;                                        ! of ODS$OPEN



                                 01FC 00000          .ENTRY  ODS$OPEN, Save R2,R3,R4,R5,R6,R7,R8          ; 0263
5E      FDF8    CE   9E 00002    MOVAB   -520(SP), SP                                                     ;
51      0200    8F   3C 00007    MOVZWL  #512, SIZE                                                       ; 0312
    00000000G   EF   16 0000C    JSB     EXE$ALONONPAGED                                                  ; 0314
08              50   E8 00012    BLBS    R0, 1$                                                           ;
50 000184D4     8F   D0 00015    MOVL    #99540, R0                                                       ;
                     04 0001C    RET                                                                      ;
```

```
                    54     A9                  52   D0 0001D  1$:    MOVL     ADDR, 84(CBLOCK)                         ; 0316
                                    4004       8F   BB 00021         PUSHR    #^M<R2,SP>                               ; 0318
                                      10       AE   9F 00025         PUSHAB   NDXHDR                                   ;
                                      48       A9   9F 00028         PUSHAB   72(CBLOCK)                               ;
                                      38       A9   DD 0002B         PUSHL    56(CBLOCK)                               ;
                 0000G     CF                  05   FB 0002E         CALLS    #5, FIL$OPENFILE                         ;
                           55                  50   E8 00033         BLBS     STAT, 6$                                 ; 0321
              00000910     8F                  50   D1 00036         CMPL     STAT, #2320                              ; 0325
                                               08   12 0003D         BNEQ     2$                                       ;
                           50  00018292        8F   D0 0003F         MOVL     #98962, R0                               ;
                                                    04 00046         RET                                               ;
              00000818     8F                  50   D1 00047  2$:    CMPL     STAT, #2072                              ; 0326
                                               08   12 0004E         BNEQ     3$                                       ;
                           50  0001852C        8F   D0 00050         MOVL     #99628, R0                               ;
                                                    04 00057         RET                                               ;
              00000808     8F                  50   D1 00058  3$:    CMPL     STAT, #2056                              ; 0327
                                               12   13 0005F         BEQL     4$                                       ;
              00000810     8F                  50   D1 00061         CMPL     STAT, #2064                              ;
                                               09   13 00068         BEQL     4$                                       ;
              000008C0     8F                  50   D1 0006A         CMPL     STAT, #2240                              ;
                                               0C   12 00071         BNEQ     5$                                       ;
                    0C     AB                  50   D0 00073  4$:    MOVL     STAT, 12(FAB)                            ;
                           50  0001C002        8F   D0 00077         MOVL     #114690, R0                              ;
                                                    04 0007E         RET                                               ;
                    0C     AB                  50   D0 0007F  5$:    MOVL     STAT, 12(FAB)                            ; 0328
                           50  0001C0F4        8F   D0 00083         MOVL     #114932, R0                              ;
                                                    04 0008A         RET                                               ; 0323
                           02        07        A2   91 0008B  6$:    CMPB     7(FILHDR), #2                            ; 0331
                                               04   12 0008F         BNEQ     7$                                       ;
                    01     A9                  04   88 00091         BISB2    #4, 1(CBLOCK)                            ; 0333
                    0C     A9      0000V       CF   9E 00095  7$:    MOVAB    ODS$GET, 12(CBLOCK)                      ; 0335
                    10     A9      0000V       CF   9E 0009B         MOVAB    ODS$READ, 16(CBLOCK)                     ; 0336
                    50     A9                  6E   D0 000A1         MOVL     STATUS, 80(CBLOCK)                       ; 0337
          08        01     A9                  02   E1 000A5         BBC      #2, 1(CBLOCK), 8$                        ; 0344
                           52                  14   C0 000AA         ADDL2    #20, R2                                  ;
                           50                  52   D0 000AD         MOVL     R2, R0                                   ;
                                               04   11 000B0         BRB      9$                                       ;
                           50        0E        A2   9E 000B2  8$:    MOVAB    14(R2), R0                               ;
                    1E     AB        01        A0   90 000B6  9$:    MOVB     1(R0), 30(FAB)                           ; 0346
       44     A9    08     A0                  10   9C 000BB         ROTL     #16, 8(R0), 68(CBLOCK)                   ; 0349
                    42     A9        0C        A0   B0 000C1         MOVW     12(R0), 66(CBLOCK)                       ; 0350
51            60           04                  04   EF 000C6         EXTZV    #4, #4, (R0), R1                         ; 0351
                    1D     AB                  51   90 000CB         MOVB     R1, 29(FAB)                              ;
51            60           04                  00   EF 000CF         EXTZV    #0, #4, (R0), R1                         ; 0352
                    1F     AB                  51   90 000D4         MOVB     R1, 31(FAB)                              ;
                    36     AB        10        A0   B0 000D8         MOVW     16(R0), 54(FAB)                          ; 0353
                                               05   12 000DD         BNEQ     10$                                      ; 0355
                    36     AB        02        A0   B0 000DF         MOVW     2(R0), 54(FAB)                           ; 0357
                    3F     AB        0F        A0   90 000E4  10$:   MOVB     15(R0), 63(FAB)                          ; 0359
                           50  00010001        8F   D0 000E9         MOVL     #65537, R0                               ; 0362
                                                    04 000F0         RET                                               ;

; Routine Size:  241 bytes,    Routine Base:  CODE + 00FF


;    0363  1
```

```
;    0364  1     %sbttl 'ODS$GET routine'
;    0365  1
;    0366  1     global routine ods$get : call_rms =
;    0367  1
;    0368  1     !++
;    0369  1     ! Functional description:
;    0370  1     !       De-block and return a record from an ODS file.
;    0371  1     !
;    0372  1     ! Inputs:
;    0373  1     !       none
;    0374  1     !
;    0375  1     ! Implicit inputs:
;    0376  1     !       R9                      CBLOCK pointer
;    0377  1     !       R10                     RAB pointer
;    0378  1     !       R11                     FAB pointer
;    0379  1     !       several fields in the above data structures
;    0380  1     !
;    0381  1     ! Outputs:
;    0382  1     !       none
;    0383  1     !
;    0384  1     ! Implicit outputs:
;    0385  1     !       RAB [rab$l_stv]         Set to error status of failing system service
;    0386  1     !       RAB [rab$l_rbf]         Set to user buffer
;    0387  1     !       RAB [rab$w_rsz]         Size of record copied to user buffer
;    0388  1     !
;    0389  1     ! Side effects:
;    0390  1     !       CBLOCK [ctl$w_rfa]      advanced one record if access was not
;    0391  1     !                               random-by-RFA
;    0392  1     !       CBLOCK                  cache control and cache buffers may be
;    0393  1     !                               altered if part or all of the record is
;    0394  1     !                               not within a block currently cached.
;    0395  1     !
;    0396  1     ! Return codes:
;    0397  1     !       RMS$_NORMAL             if OK
;    0398  1     !       RMS$_RER                if system read error (QIO)
;    0399  1     !       RMS$_RFA                if random RFA is beyond EOF
;    0400  1     !       RMS$_EOF                if RFA is beyond end of file
;    0401  1     !       RMS$_IRC                Illegal record
;    0402  1     !       RMS$_RTB                Record is too big for user buffer (truncated)
;    0403  1     !--
;    0404  1
;    0405  2         begin
;    0406  2
;    0407  2         external register
;    0408  2             cblock = 9 : ref bblock,            ! Pointer to CBLOCK
;    0409  2             rab = 10 : ref bblock,              ! Pointer to RAB
;    0410  2             fab = 11 : ref bblock;              ! Pointer to FAB
;    0411  2
;    0412  2         global register
;    0413  2             block = 6 : ref vector [, byte],    ! Pointer to cache buffer
;    0414  2             vbn = 7,                            ! Current VBN in file
;    0415  2             offset = 8;                         ! Current offset in VBN
;    0416  2
;    0417  2         local
;    0418  2             f_size : word,                      ! Full size of record                [04]
;    0419  2             size : word,                        ! Size of record                     [04]
;    0420  2             lenlim : word,                      ! Size of user buffer                [04]
```

```
;    0421  2              ptr;                                    ! Address in user buffer
;    0422  2
;    0423  2          vbn = .cblock [ctl$l_vbn];                  ! Extract vbn from next RFA
;    0424  2          offset = .cblock [ctl$w_byte];              ! Extract byte offset
;    0425  2
;    0426  2          do
;    0427  3              begin                                   ! Loop until find real record
;    0428  3
;    0429  3              if .rab [rab$b_rac] neq rab$c_rfa       ! If not random access
;    0430  3              then
;    0431  3                  ods$advance (.cblock [ctl$w_rec_size]);     ! Else advance to next
;    0432  3
;    0433  3              cblock [ctl$l_vbn] = .vbn;              ! Update CBLOCK vbn
;    0434  3              cblock [ctl$w_byte] = .offset;          !  and it's offset
;    0435  3
;    0436  3              if .vbn gtr .cblock [ctl$l_eofvbn] or .vbn eql .cblock [ctl$l_eofvbn] and .offset geq .cblock [
;    0437  3                      ctl$w_offbyte]                  ! If past file EOF
;    0438  3              then
;    0439  3                  return rms$_eof;                    ! can't read past EOF
;    0440  3
;    0441  4              (
;    0442  4
;    0443  4              local
;    0444  4                  stat;
;    0445  4
;    0446  4              stat = ods$access ();                   ! Load the block in cache
;    0447  3              if not .stat then return .stat);
;    0448  3              cblock [ctl$w_rec_size] =               ! Set size of current record
;    0449  4              (if .fab [fab$b_rfm] neq fab$c_fix      ! If not fixed record format
;    0450  4              then
;    0451  5                  begin
;    0452  5                  size = .(block [.offset])<0, 16>;   ! then fetch record size
;    0453  5                  ods$advance (2);                    ! Skip over the size
;    0454  6                  (
;    0455  6
;    0456  6                  local
;    0457  6                      stat;
;    0458  6
;    0459  6                  stat = ods$access ();               ! and be sure (new) VBN is there
;    0460  5                  if not .stat then return .stat);
;    0461  5                  .size + 2
;    0462  5                  end
;    0463  4              else size = .fab [fab$w_mrs])           ! else get fixed size from FAB
;    0464  3              end
;    0465  3          until (if .size eql %x'FFFF'                ! Means non-spanned 'skip to next block'
;    0466  3              then
;    0467  4                  begin
;    0468  4                  vbn = .vbn + 1;
;    0469  4                  cblock [ctl$w_rec_size] = offset = 0;
;    0470  4                  0
;    0471  4                  end
;    0472  2              else 1);
;    0473  2
;    0474  2          if .fab [fab$b_rfm] eql fab$c_vfc           ! If we have header record
;    0475  2          then
;    0476  3              begin
;    0477  3
```

```
;    0478  3              if .size lss .fab [fab$b_fsz]
;    0479  3              then
;    0480  4                  (rab [rab$l_stv] = .vbn;                ! Store VBN of illegal record
;    0481  4                  return rms$_irc                         ! Illegal record
;    0482  3                  );
;    0483  3
;    0484  3              if .rab [rab$l_rhb] neq 0                   ! If there's a place to put it
;    0485  3              then
;    0486  3                  ch$move (.fab [fab$b_fsz], block [.offset], .rab [rab$l_rhb]);      ! copy it over
;    0487  3
;    0488  3              size = .size - .fab [fab$b_fsz];            ! Decrease record size by header length
;    0489  3              ods$advance (.fab [fab$b_fsz])              ! and advance over it anyway
;    0490  2              end;
;    0491  2
;    0492  2          ptr = rab [rab$l_rbf] = .rab [rab$l_ubf];       ! Point to user buffer
;    0493  2          lenlim = .rab [rab$w_usz];                      ! Size limit
;    0494  2          rab [rab$w_rsz] = 0;                            ! Init output size
;    0495  2          f_size = .size;                                 ! Store size, since SIZE moves
;    0496  2
;    0497  2          while 1 do                                      ! Copy over the record
;    0498  3              begin
;    0499  3
;    0500  3              local
;    0501  3                  length;
;    0502  3
;    0503  4              (
;    0504  4
;    0505  4              local
;    0506  4                  stat;
;    0507  4
;    0508  4              stat = ods$access ();                       ! Be sure VBN is there
;    0509  3              if not .stat then return .stat);
;    0510  3              length = min (.size, (512 - .offset), .lenlim); ! Size to copy
;    0511  3              ptr = ch$move (.length, block [.offset], .ptr); ! Copy (part of) record
;    0512  3              rab [rab$w_rsz] = .rab [rab$w_rsz] + .length;   ! Update actual length so far
;    0513  3              size = .size - .length;                     ! Rest of record
;    0514  3
;    0515  3              if .size eql 0 then return rms$_normal;     ! Return if done
;    0516  3
;    0517  3              lenlim = .lenlim - .length;                 ! Buffer size left
;    0518  3
;    0519  3              if .lenlim eql 0                            ! If no space left
;    0520  3              then
;    0521  4                  begin
;    0522  4                  rab [rab$l_stv] = .f_size;              ! Return true size
;    0523  4                  return rms$_rtb
;    0524  3                  end;
;    0525  3
;    0526  3              ods$advance (.length)                       ! Skip on to next block
;    0527  2              end;
;    0528  2
;    0529  2          rms$_normal
;    0530  1          end;
```

```
                                                01FC 00000          .ENTRY  ODS$GET, Save R2,R3,R4,R5,R6,R7,R8          ; 0366
                                             5E   10 C2 00002       SUBL2   #16, SP                                     ;
                                             57 18 A9 D0 00005      MOVL    24(CBLOCK), VBN                             ; 0423
                                             58 1C A9 3C 00009      MOVZWL  28(CBLOCK), OFFSET                          ; 0424
                                             02 1E AA 91 0000D 1$:  CMPB    30(RAB), #2                                 ; 0429
                                                07 13 00011         BEQL    2$                                          ;
                                             50 1E A9 3C 00013      MOVZWL  30(CBLOCK), R0                              ; 0431
                                                FDF6 30 00017       BSBW    UDS$ADVANCE                                 ;
                                          18 A9    57 D0 0001A 2$:  MOVL    VBN, 24(CBLOCK)                             ; 0433
                                          1C A9    58 B0 0001E      MOVW    OFFSET, 28(CBLOCK)                          ; 0434
                                          44 A9    57 D1 00022      CMPL    VBN, 68(CBLOCK)                             ; 0436
                                                   0A 14 00026      BGTR    3$                                          ;
                                                   10 12 00028      BNEQ    4$                                          ;
     58    42 A9                          10       00 ED 0002A      CMPZV   #0, #16, 66(CBLOCK), OFFSET                 ;
                                                   08 14 00030      BGTR    4$                                          ;
                                             50 0001827A   8F D0 00032 3$:  MOVL    #98938, R0                          ; 0439
                                                   04 00039         RET                                                 ;
                                                FDEB 30 0003A 4$:   BSBW    ODS$ACCESS                                  ; 0446
                                             15    50 E9 0003D      BLBC    STAT, 5$                                    ; 0447
                                             01 1F AB 91 00040      CMPB    31(FAB), #1                                 ; 0449
                                                1A 13 00044         BEQL    6$                                          ;
                                                6846 9F 00046       PUSHAB  (OFFSET)[BLOCK]                             ; 0452
                                             6E    9E B0 00049      MOVW    @(SP)+, SIZE                                ;
                                             50    02 D0 0004C      MOVL    #2, R0                                      ; 0453
                                                FDBE 30 0004F       BSBW    ODS$ADVANCE                                 ;
                                                FDD3 30 00052       BSBW    ODS$ACCESS                                  ; 0459
                                             75    50 E9 00055 5$:  BLBC    STAT, 13$                                   ; 0460
                                             50    6E 3C 00058      MOVZWL  SIZE, R0                                    ; 0461
                                             50    02 C0 0005B      ADDL2   #2, R0                                      ;
                                                   07 11 0005E      BRB     7$                                          ;
                                             50 36 AB 3C 00060 6$:  MOVZWL  54(FAB), R0                                 ; 0463
                                             6E    50 B0 00064      MOVW    R0, SIZE                                    ;
                                          1E A9    50 B0 00067 7$:  MOVW    R0, 30(CBLOCK)                              ; 0469
                                       FFFF 8F     6E B1 0006B      CMPW    SIZE, #65535                                ; 0465
                                                   09 12 00070      BNEQ    8$                                          ;
                                                   57 D6 00072      INCL    VBN                                         ; 0468
                                                   58 D4 00074      CLRL    OFFSET                                      ; 0469
                                                1E A9 B4 00076      CLRW    30(CBLOCK)                                  ;
                                                   92 11 00079      BRB     1$                                          ;
                                             03 1F AB 91 0007B 8$:  CMPB    31(FAB), #3                                 ; 0474
                                                   32 12 0007F      BNEQ    11$                                         ;
                                             50 3F AB 9A 00081      MOVZBL  63(FAB), R0                                 ; 0478
                                             6E    50 B1 00085      CMPW    R0, SIZE                                    ;
                                                   0C 1B 00088      BLEQU   9$                                          ;
                                          0C AA    57 D0 0008A      MOVL    VBN, 12(RAB)                                ; 0480
                                             50 0001857C   8F D0 0008E MOVL    #99708, R0                              ; 0481
                                                   04 00095         RET                                                 ;
                                                2C AA D5 00096 9$:  TSTL    44(RAB)                                     ; 0484
                                                   0A 13 00099      BEQL    10$                                         ;
                                             50 3F AB 9A 0009B      MOVZBL  63(FAB), R0                                 ; 0486
  2C  BA                                  6846    50 28 0009F      MOVC3   R0, (OFFSET)[BLOCK], @44(RAB)               ;
                                             50 3F AB 9A 000A5 10$: MOVZBL  63(FAB), R0                                 ; 0488
                                             6E    50 A2 000A9      SUBW2   R0, SIZE                                    ;
                                             50 3F AB 9A 000AC      MOVZBL  63(FAB), R0                                 ; 0489
                                                FD5D 30 000B0       BSBW    ODS$ADVANCE                                 ;
                                             50 24 AA D0 000B3 11$: MOVL    36(RAB), R0                                 ; 0492
```

```
                                28   AA            50   D0 000B7          MOVL    R0, 40(RAB)                                          ;
                                53                 50   D0 000BB          MOVL    R0, PTR                                              ;
                                08   AE       20   AA   B0 000BE          MOVW    32(RAB), LENLIM                                      ; 0493
                                           22   AA   B4 000C3          CLRW    34(RAB)                                              ; 0494
                                0C   AE            6E   B0 000C6          MOVW    SIZE, F_SIZE                                         ; 0495
                                              FD5B   30 000CA 12$:       BSBW    ODS$ACCESS                                           ; 0508
                                58                 50   E9 000CD 13$:       BLBC    STAT, 18$                                            ; 0509
                     51 00000200   8F            58   C3 000D0          SUBL3   OFFSET, #512, R1                                     ; 0510
                                50                 6E   3C 000D8          MOVZWL  SIZE, R0                                             ;
                                51                 50   D1 000DB          CMPL    R0, R1                                               ;
                                                   03   15 000DE          BLEQ    14$                                                  ;
                                50                 51   D0 000E0          MOVL    R1, R0                                               ;
          50    08   AE          10                 00   ED 000E3 14$:       CMPZV   #0, #16, LENLIM, R0                                  ;
                                                   04   18 000E9          BGEQ    15$                                                  ;
                                50         08   AE   3C 000EB          MOVZWL  LENLIM, R0                                           ;
                                04   AE            50   D0 000EF 15$:       MOVL    R0, LENGTH                                           ;
                     63               6846       04   AE   28 000F3          MOVC3   LENGTH, (OFFSET)[BLOCK], (PTR)               ; 0511
                                22   AA            04   AE   A0 000F9          ADDW2   LENGTH, 34(RAB)                              ; 0512
                                6E                 04   AE   A2 000FE          SUBW2   LENGTH, SIZE                                 ; 0513
                                                   1D   13 00102          BEQL    17$                                          ; 0515
                                08   AE            04   AE   A2 00104          SUBW2   LENGTH, LENLIM                               ; 0517
                                                   0D   12 00109          BNEQ    16$                                          ; 0519
                                0C   AA            0C   AE   3C 0010B          MOVZWL  F_SIZE, 12(RAR)                              ; 0522
                                50   000181A8     8F   D0 00110          MOVL    #98728, R0                                   ; 0523
                                                        04 00117          RET                                                  ;
                                50                 04   AE   D0 00118 16$:       MOVL    LENGTH, R0                                   ; 0526
                                                  FCF1   30 0011C          BSBW    ODS$ADVANCE                                  ;
                                                   A9   11 0011F          BRB     12$                                          ;
                                50   00010001      8F   D0 00121 17$:       MOVL    #65537, R0                                   ; 0530
                                                        04 00128 18$:       RET                                                  ;

; Routine Size:  297 bytes,    Routine Base:  CODE + 01F0

;   0531  1
```

```
;    0532  1     %sbttl 'ODS$READ routine'
;    0533  1
;    0534  1     global routine ods$read : call_rms =
;    0535  1
;    0536  1     !++
;    0537  1     ! Functional description:
;    0538  1     !       Read virtual blocks of file
;    0539  1     !
;    0540  1     ! Inputs:
;    0541  1     !       none
;    0542  1     !
;    0543  1     ! Implicit inputs:
;    0544  1     !       R9                      CBLOCK pointer
;    0545  1     !       R10                     RAB pointer
;    0546  1     !       various fields in the above-listed data structures
;    0547  1     !
;    0548  1     ! Outputs:
;    0549  1     !       none
;    0550  1     !
;    0551  1     ! Implicit outputs:
;    0552  1     !       RAB [rab$l_rbf]         points to buffer
;    0553  1     !       RAB [rab$w_rsz]         size of buffer
;    0554  1     !
;    0555  1     ! Side Effects:
;    0556  1     !       CBLOCK [ctl$l_nbp]      points to next VBN for sequential READ.
;    0557  1     !
;    0558  1     ! Return codes:
;    0559  1     !       RMS$_NORMAL             OK
;    0560  1     !       RMS$_RER                Read error
;    0561  1     !--
;    0562  1
;    0563  2         begin
;    0564  2
;    0565  2         external register
;    0566  2             cblock = 9 : ref bblock,                ! Map the control block
;    0567  2             rab = 10 : ref bblock;                  ! Map the RAB
;    0568  2
;    0569  2         local
;    0570  2             size,
;    0571  2             next;
;    0572  2
;    0573  2         next = .cblock [ctl$l_nbp];                 ! Block to start at
;    0574  2         size = rab [rab$w_rsz] = min (.rab [rab$w_usz],     ! Determine actual size to
;    0575  2             (.cblock [ctl$l_eofvbn] - .next)*512 +  !  read--either given size
;    0576  2             .cblock [ctl$w_offbyte]);               !  or to end of file if less
;    0577  2         cblock [ctl$l_nbp] = .next + (.size + 511)/512;     ! Point to next full block
;    0578  2
;    0579  2         if .cblock [ctl$l_startlbn] neq 0           ! If file is contiguous
;    0580  2         then
;    0581  3             begin
;    0582  3
;    0583  3             local
;    0584  3                 stat;
;    0585  3
;    0586  3             stat = boo$qio (rab [rab$l_rbf] = .rab [rab$l_ubf], .size, .cblock [ctl$l_startlbn] + .next - 1,
;    0587  3                 io$_readlblk, 0, .cblock [ctl$l_rpb]);      ! Read all at once
;    0588  3
```

```
;  3              if not .stat                          ! If error, scram
;  3              then
;  4                  (rab [rab$l_stv] = .stat; return rms$_rer)
;  4
;  3              end
;  2          else                                      ! Non-contiguous, read it
;  3              begin                                 !  block by block
;  3
;  3              local
;  3                  ptr;
;  3
;  3              ptr = rab [rab$l_rbf] = .rab [rab$l_ubf];        ! Start of user buffer
;  3
;  3              while .size gtr 0 do
;  4                  begin
;  4
;  4                  global register
;  4                      block = 6,                    ! Registers for ACCESS
;  4                      vbn = 7,
;  4                      offset = 8;
;  4
;  4                  local
;  4                      rs,
;  4                      stat;
;  4
;  4                  vbn = .next;                      ! Next block to read
;  4                  rs = min (.size, 512);            ! Read a block or less
;  4                  stat = ods$access ();             ! Access next block
;  4
;  4                  if not .stat                      ! Scram if error
;  4                  then
;  4                      (rab [rab$l_stv] = .stat; return rms$_rer);
;  4
;  4                  ptr = ch$move (.rs, .block, .ptr);  ! Copy to user buffer
;  4                  next = .next + 1;                 ! Advance block number
;  4                  size = .size - .rs                ! Count size of read
;  4                  end
;  4
;  2              end;
;  2
;  2          rms$_normal
;  1          end;                                      ! Of ods$read


                                 09FC 00000      .ENTRY  ODS$READ, Save R2,R3,R4,R5,R6,R7,R8,R11      ; 0534
                    5E         04   C2 00002      SUBL2   #4, SP                                       ;
                           04  A9   DD 00005      PUSHL   4(CBLOCK)                                    ; 0573
     51       44    A9         6E   C3 00008      SUBL3   NEXT, 68(CBLOCK), R1                         ; 0575
     51             51         09   78 0000D      ASHL    #9, R1, R1                                   ;
                    50     42  A9   3C 00011      MOVZWL  66(CBLOCK), R0                               ; 0576
                    51         50   C0 00015      ADDL2   R0, R1                                       ;
                    50     20  AA   3C 00018      MOVZWL  32(RAB), R0                                  ; 0575
                    51         50   D1 0001C      CMPL    R0, R1                                       ;
                               03   15 0001F      BLEQ    1$                                           ;
```

```
                                50              51  D0 00021          MOVL    R1, R0                                  ; 0574
                             22 AA              50  B0 00024 1$:      MOVW    RC, 34(RAB)
                                5B              50  D0 00028          MOVL    R0, SIZE                                ; 0577
                                50        01FF  CB  9E 0002B          MOVAB   511(R11), R0                            ; 0579
                                50 00000200   8F  C6 00030          DIVL2   #512, R0                                ; 0587
                          04    A9            50  6E C1 00037          ADDL3   NEXT, R0, 4(CBLOCK)                     ; 0586
                                         50   A9  D5 0003C          TSTL    80(CBLOCK)
                                              24  13 0003F          BEQL    2$
                                         38   A9  DD 00041          PUSHL   56(CBLOCK)
                                7E            21  7D 00044          MOVQ    #33, -(SP)
                          50    50   A9       0C  AE C1 00047          ADDL3   NEXT, 80(CBLOCK), R0
                                             FF   A0  9F 0004D          PUSHAB  -1(R0)
                                5B            DD  00050          PUSHL   SIZE
                                         24   AA  DD 00052          PUSHL   36(RAB)
                             28 AA              6E  D0 00055          MOVL    (SP), 40(RAB)
                      00000000G EF              06  FB 00059          CALLS   #6, B00$QIO
                                48              50  E8 00060          BLBS    STAT, 7$                                ; 0589
                                                2D  11 00063          BRB     5$                                      ; 0591
                                50        24   AA  D0 00065 2$:      MOVL    36(RAB), R0                             ; 0600
                             28 AA              50  D0 00069          MOVL    R0, 40(RAB)
                                53              50  D0 0006D          MOVL    R0, PTR
                                                5B  D5 00070 3$:      TSTL    SIZE                                    ; 0602
                                                37  15 00072          BLEQ    7$
                                57              6E  D0 00074          MOVL    NEXT, VBN                               ; 0614
                                50              5B  D0 00077          MOVL    SIZE, R0                                ; 0615
                      00000200  8F              50  D1 0007A          CMPL    R0, #512
                                                05  15 00081          BLEQ    4$
                                50        0200  8F  3C 00083          MOVZWL  #512, R0
                             04 AE              50  D0 00088 4$:      MOVL    R0, RS
                                           FC70  30  0008C          BSBW    ODS$ACCESS                              ; 0616
                                0C              50  E8 0008F          BLBS    STAT, 6$                                ; 0618
                             0C AA              50  D0 00092 5$:      MOVL    STAT, 12(RAB)                           ; 0620
                                50 0001C0F4   8F  D0 00096          MOVL    #114932, R0
                                                04  0009D          RET
                          63                66        04  AE 28 0009E 6$:      MOVC3   RS, (BLOCK), (PTR)              ; 0622
                                                    6E  D6 000A3          INCL    NEXT                                ; 0623
                                5B            04  AE  C2 000A5          SUBL2   RS, SIZE                            ; 0624
                                                C5  11 000A9          BRB     3$
                                50 00010001   8F  D0 000AB 7$:      MOVL    #65537, R0                              ; 0630
                                                04 000B2          RET

; Routine Size:  179 bytes,    Routine Base:  CODE + 0319

;   0631  1
;   0632  1      end
;   0633  1
;   0634  0      eludom



;                                   PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
```

```
;
;  CODE                          972  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DRB1:[DS.WORK]DIAG.L32;265	784	0	0	85	00:00.1
DRB1:[DS.WORK]DS.L32;159	653	21	3	42	00:00.2
SYS$SYSROOT:[SYSLIB]LIB.L32;7	18017	43	0	975	00:04.7

COMMAND QUALIFIERS

```
;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE ODS

; Size:          972 code + 0 data bytes
; Run Time:        00:22.2
; Elapsed Time:    00:27.7
; Lines/CPU Min:    1716
; Lexemes/CPU-Min: 13481
; Memory Used:  151 pages
; Compilation Complete
```

Table of contents

(1)	74	Macro invocations	
(2)	136	DSP$CONFIG_ADP	Build P-table for adapter
(5)	251	MAP$IOSPACE	Validate mapping for adapters
(7)	303	FETCH_LONG	Retreive longword from possibly bad address
(8)	348	PURGE_DATAPATH	
(10)	391	UBA$INITIAL	UNIBUS ADAPTER INITIALIZATION
(11)	419	IO$TESTCSR_x	Test CSR for existence
(12)	468	IOGEN$CONN_VEC	
(13)	495	QIOCLEAN_CPU	
(14)	517	ONLY_SCB	CPU specific SCB setup
(15)	528	SCBVECTOR_xxx	Otherwise undefined SCB vectors
(15)	580	ONLY_CLOCK_INI'	Perform cpu-specific TODR initialization

```
0000     1          .TITLE  ONLY730 *** ONLY730 NEBULA specific routines
0000     2          .IDENT /07-12/
0000     3          .LIST   MEB
0000     4          .NLIST  CND
0000     5          .DSABL  GBL
0000     6
0000     7  ;
0000     8  ; Copyright (c) 1979, 1983, 1984
0000     9  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    10  ;
0000    11  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    12  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    13  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    14  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    15  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    16  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    17  ; REMAIN IN DEC.
0000    18  ;
0000    19  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    20  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    21  ; CORPORATION.
0000    22  ;
0000    23  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    24  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    25
0000    26  ;++
0000    27  ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    28  ;
0000    29  ; ABSTRACT:
0000    30  ;
0000    31  ; ENVIRONMENT:
0000    32  ;
0000    33  ; AUTHOR:       Roger Riggs     20-FEB-79       VERSION 01.
0000    34  ;
0000    35  ; MODIFIED BY:
0000    36  ;
0000    37  ;                Roger Riggs, 24-Jan-1989, Version 5.2
0000    38  ;       02       Added definition of value DS$GK_USOVR, for
0000    39  ;                WAITUS clock overhead.
0000    40  ;                Roger Riggs, 15-Feb-1980, Version 5.2
0000    41  ;       03       Fixed QIOCLEAN_CPU to correctly restore the SCB
0000    42  ;                Dave Butenhof, 07-May-1981, version 6.4
0000    43  ;       04       Fix PURGDATAP routine to set correct address for UBA map.
0000    44  ;       05       ATTACH DW730, not DW750.
0000    45  ;
0000    46  ;       06       - Dave Butenhof, 23-Nov-1981, version 6.5
0000    47  ;                Add .LIBRARY declarations, change SEP Psect to Code and
0000    48  ;                Work.
0000    49  ;
0000    50  ;       07       - Dave Butenhof, 09-Mar-1982, version 6.6
0000    51  ;                Fix rruncation errors.
0000    52  ;
0000    53  ;       08       Bob Bergazzi    April 26, 1983  Version 6.11
0000    54  ;                Added ONLY_SCB routine for 11/xxx. Does nothing for NEBULA.
0000    55  ;
0000    56  ;       09       Bob Bergazzi    May 16, 1983    Version 6.11
0000    57  ;                Changed the order of the .LIB statements.
```

```
                                58 ;
                                59 ;       10      Bob Bergazzi 11-16-83   Version 6.14
                                60 ;               Changed MODNAM from ONLY750 to ONY730.
                                61 ;
                                62 ;       11      Bob Bergazzi    Jan. 25, 1984   Version 6.14
                                63 ;               Added ONLY_CLOCK_INIT routine, which reads the TODR.
                                64 ;               We no longer want to MFPR the TODR in CLK$RE_INIT
                                65 ;               because not all cpus read the TODR with an MFPR.
                                66 ;               Also, TODR is now defined by $PR730DEF as PR730$_TODR.
                                67 ;
                                68 ;       12      Bob Bergazzi    May 9, 1984     Version 7.0
                                69 ;               Added UBINTSZ in order to remove code from QIO.
                                70 ;--
                                71
                                72
```

```
                                0000     74 .subtitle       Macro invocations
                                0000     75
                                0000     76 .library        /sys$library:lib/       ;                                [09]
                                0000     77 .library        /$DS/                   ;                                [09]
                                0000     78 .library        /$DIAG/                 ;                                [09]
                                0000     79
                                0000     80         .NLIST  ME,MEB
                                0000     81        $UCBDEF
                                0000     82        $CRBDEF
                                0000     83        $ADPDEF
                                0000     84        $VECDEF
                                0000     85        $DYNDEF
                                0000     86        $PSLDEF
                                0000     87        $PTEDEF
                                C000     88        $VIELD  PTE,<29-9>,<<10,,M>>
                                0000     89        $PRTDEF
                                0000     90        $PRDEF
                                0000     91        $PR730DEF                       ; TODR now here for VMS v4       [11]
                                0000     92        $DS_HPODEF
                                0000     93        $DS_DW730_DEF                   ;                                [05]
                                0000     94        $DS_DSDEF
                                0000     95        $UBIDEF
                                0000     96
                    00000001    0000     97 SS$_NORMAL=1
                    00000001    0000     98 IS=1
                                0000     99
                                0000    100 ;
                                0000    101 ; Clock overhead factor
                                0000    102 ;
                    00000200    0000    103 DS$GK_USOVR     == ^X200                ; Overflow factor for 11/730
                                0000    104 ;
                                0000    105 ; I/O SPACE DEFINITION
                                0000    106 ;
                                0000    107
                    40000000    0000    108 IOC$K_IOSPACE == ^X40000000
                    03000000    0000    109 DS$GK_SID       ==      3@24            ; Pseudo SID for 11/730
                                0000    110
                                0000    111         .EXTRN  BUG$CHECK,      CMK$RCONSOLE, CMK$TCONSOLE,     DS$WAITUS
                                0000    112         .EXTRN  DS_ERRSUP
                                0000    113         .EXTRN  EXE_UNEXPINT
                                0000    114         .EXTRN  EXE$ALONONPAGED, EXE$DEANONPAGED,       FETCH_STORE
                                0000    115         .EXTRN  INS$PTABLE,     IOC$GL_ADPLIST
                                0000    116         .EXTRN  MAPMEM$IOSPACE
                                0000    117         .EXTRN  QIO$DEALLOCATE
                                0000    118         .EXTRN  SCB_BASE, SCB_IMAGE
                                0000    119         .EXTRN  SS$_CONTINUE,   SS$_CONTROLC,   SS$_CTRLERR,    SS$_DATACHECK
                                0000    120         .EXTRN  SS$_RESIGNAL
                                0000    121         .EXTRN  SYS$SETPRT,     SYS$UNWIND
                                0000    122         .EXTRN  TST$MCHK
                                0000    123         .EXTRN  UBINT_DISP
                                0000    124         .EXTRN  SCB_UNKINT
                                0000    125
                          00000000    126 .Psect  Work, Noshr, Noexe, Wrt, Long
                                0000    127
                                0000    128 LOCAL_STORE:
                    00000000    0000    129         .LONG   0
                                0004    130
```

```
                    00000000   131 .Psect  Data, Shr, Noexe, Nowrt, Long
                        0000   132         MODNAM  ONLY730
                        0008   133 IO$GQ_PHYSICAL::
01000000 00F00000       0008   134         .LONG   ^XF00000,^X1000000
```

```
                        0010   136          .SBTTL  DSP$CONFIG_ADP  Build P-table for adapter
                        0010   137 ;++
                        0010   138 ; FUNCTIONAL DESCRIPTION:
                        0010   139 ;
                        0010   140 ;       This routine builds and inserts a P-table for
                        0010   141 ;       the channel/adapter addressed.
                        0010   142 ;
                        0010   143 ; CALLING SEQUENCE:
                        0010   144 ;
                        0010   145 ;       DSP$CONFIG_ADP(Adapter_address,PT_address,type_address)
                        0010   146 ;
                        0010   147 ; INPUT PARAMETERS:
                        0010   148 ;
                        0010   149 ;       4(AP)   Address of configuration status register of adapter
                        0010   150 ;
                        0010   151 ; IMPLICIT INPUTS:      NONE
                        0010   152 ;
                        0010   153 ; OUTPUT PARAMETERS:
                        0010   154 ;
                        0010   155 ;       8(AP)   Address to store address of P-table created
                        0010   156 ;       12(AP)  Address to store type of adapter, 0=MBA, 1=UBA
                        0010   157 ;
                        0010   158 ; IMPLICIT OUTPUTS:
                        0010   159 ;
                        0010   160 ;       P-table for Adapter
                        0010   161 ;
                        0010   162 ; COMPLETION CODES:
                        0010   163 ;
                        0010   164 ;       SS$_NORMAL
                        0010   165 ;       DS$_MACHCK
                        0010   166 ;
                        0010   167 ; SIDE EFFECTS:
                        0010   168 ;
                        0010   169 ;       The adapter specified is pinged to determine
                        0010   170 ;       interrupt level and interrupt vector.
                        0010   171 ;
                        0010   172 ; REGISTER USAGE:
                        0010   173 ;
                        0010   174 ;       R4      Address of current PT
                        0010   175 ;       R5      Address of channel
                        0010   176 ;--
                        0010   177          .NLIST  ME,MEB
                        0010   178          $OFFSET 0,NEGATIVE, < -
                        0010   179                  <L_VECTOR,4>, -
                        0010   180                  <L_IPL,4>, -
                        0010   181                  <L_CONFIG,4>, -
                        0010   182                  <L_LOCAL,0>>
                        FFFC       L_VECTOR:
                        FFF8       L_IPL:
                        FFF4       L_CONFIG:
                        FFF4       L_LOCAL:
                        0010   183
                        0010   184          .LIST   MEB
```

```
                                00000000   186 .Psect  Code, Shr, Exe, Nowrt, Long
                                    0000   187
                                001C 0000   188 .ENTRY  DSP$CONFIG_ADP,^M<R2,R3,R4>
                                    0002   189
                                    0002   190 ;       MOVAB   W^HANDLR,(FP)           ; Exception handler
                          5E   F4 AD   9E   0002   191         MOVAB   L_LOCAL(FP),SP          ; Allocate space for local storage
                00000000'EF    5D   D0   0006   192         MOVL    FP,L^LOCAL_STORE        ; And save address for interrupts       [07]
                           FC AD   D4   000D   193         CLRL    L_VECTOR(FP)            ; Clear vector
                           F8 AD   D4   0010   194         CLRL    L_IPL(FP)               ; Clear IPL
                                    0013   195
      54   04 AC    40000000 8F   C9   0013   196         BISL3   #IOC$K_IOSPACE,4(AP),R4 ; Base adapter
                        F4 AD   54   D0   001C   197         MOVL    R4,L_CONFIG(FP)         ; Store Configuration register addr
                           50   64   D0   0020   198         MOVL    (R4),R0                 ; Get Configuration register
            50   50   05   03   EF   0023   199         EXTZV   #3,#5,R0,R0             ; Get adapter type
                                    C028   200         CASE    R0,LIMIT=#5,DISPLIST=<40$>; UBI only
                 00'   05   50   AF   0028             CASEW   R0,#5,S^#<<30001$-30000$>/2>-1
                                    002C        30000$:
                                001B' 002C             .SIGNED_WORD    40$-30000$
                                    002E        30001$:
                                    002E   201         ERRSUP_S                        ; Unknown adapter type
                     00000000'EF    DF   002E                   PUSHAL  $MODULE
                                 00   DD   0034                   PUSHL   #0
                                 01   DD   0036                   PUSHL   #$ER
                00000000'EF    03   FB   0038                   CALLS   #3, DS_ERRSUP
                50   00660002 8F   D0   003F   202         MOVL    #DS$_ERROR,R0           ; Error
                                 04   0046   203         RET
                                    0047   204
                      0C BC    01   D0   0047   205 40$:   MOVL    #1,@12(AP)              ; Set adapter type to UBA
                         51    32   D0   004B   206         MOVL    #DW730$K_LEN,R1         ; Set length of PT                      [05]
                            FFAF'   30   004E   207         BSBW    EXE$ALONONPAGED         ; Allocate space for PT
                              04 50   E8   0051   208         BLBS    R0,60$                  ; Branch if successful
                         5E    08   C0   0054   209         ADDL    #8,SP                   ; Reset stack
                                 04   0057   210 50$:   RET                             ; Return on error
                                    0058   211
      1C A2    40FC0000 8F   D0   0058   212 60$:   MOVL    #^X40FC0000,HP$A_DVA(R2); Set device virtual address
                        08 A2   51   B0   0060   213         MOVW    R1,HP$W_SIZE(R2)        ; Set length of PT
      0C A2    3057445F 8F   D0   0064   214         MOVL    #^A''_DW0'',HP$T_DEVICE(R2); Set device name
         26 A2    4405 8F   B0   006C   215         MOVW    #^X4405,HP$T_TYPE(R2)   ; Set length and ''D'' of ''DW750''
      28 A2    30333757 8F   D0   0072   216         MOVL    #^A'W730'',HP$T_TYPE+2(R2); Set rest of ''W730''                [05]
         24 A2    0200 8F   B0   007A   217         MOVW    #^X200,HP$W_VECTOR(R2)  ; Set SCB base for UBI0
                                    0080   218
                         62    04   D0   0080   219 70$:   MOVL    #4,HP$Q_DEVICE(R2)      ; Set length of device name
                04 A2    0C A2   9E   0083   220         MOVAB   HP$T_DEVICE(R2), -
                                    0088   221                 HP$Q_DEVICE+4(R2)       ; Set address of device name
                      18 A2    54   D0   0088   222         MOVL    R4,HP$A_DEVICE(R2)      ; Set address of device
                                    008C   223
                         20 A2    D4   008C   224         CLRL    HP$A_LINK(R2)           ; Clear address of link PT
                      08 BC    52   D0   008F   225         MOVL    R2,@8(AP)               ; Store address of PT just made
                00000000'EF    62   FA   0093   226         CALLG   (R2),INS$PTABLE         ; Insert this in the table
                           0D 50   E8   009A   227         BLBS    R0,CONFIG_EXIT          ; Exit complete
                                    009D   228
                                    009D   229 CONFIG_RELEASE:
                         50    52   D0   009D   230         MOVL    R2,R0                   ; Copy address of buffer
                              FF5D'   30   00A0   231         BSBW    EXE$DEANONPAGED         ; Deallocate buffer
                50   00660002 8F   D0   00A3   232         MOVL    #DS$_ERROR,R0           ; Flag error
                                    00AA   233 CONFIG_EXIT:
                                 04   00AA   234         RET
```

```
                          0004  00AB   236 HANDLR: .WORD    ^M<R2>                      ; Unexpected exception routine
      52   00000000'EF      D0  00AD   237          MOVL    L^LOCAL_STORE,R2            ; Base stack                                [07]
        51    04 BC         DE  00B4   238          MOVAL   @4(AP),R1                   ; Point to signal arguments
     50   0000'8F           3C  00B8   239          MOVZWL  #SS$_RESIGNAL,R0            ; Assume not ours
                                00BD   240
04 A1   006600D8 8F           D1  00BD   241          CMPL    #DS$_UNEXPINT,4(R1)         ; Unexpected interrupt?
                 14           12  00C5   242          BNEQ    10$                         ; Branch if not
      FC A2    08 A1         D0  00C7   243          MOVL    8(R1),L_VECTOR(R2)          ; Store interrupt vector
        F8 A2    12          DB  00CC   244          MFPR    #PR$_IPL,L_IPL(R2)          ; Save current IPL
        50    F4 A2          D0  00D0   245          MOVL    L_CONFIG(R2),R0             ; Get Configuration register address
   50   00000000'8F           D0  00D4   246          MOVL    #SS$_CONTINUE,R0            ; Continue from exception
                                00DB   247
                          04    00DB   248 10$:     RET
                                00DC   249
```

```
                                00DC   251          .SBTTL  MAP$IOSPACE     Validate mapping for adapters
                                00DC   252 ;++
                                00DC   253 ; FUNCTIONAL DESCRIPTION:
                                00DC   254 ;
                                00DC   255 ;       This is routine is called from MAPMEM (MEMMGT) to
                                00DC   256 ;       validate those P1 space address that map thru to physical
                                00DC   257 ;       I/O space... bit 29=1
                                00DC   258 ;       For each slot, 16 pages are mapped.
                                00DC   259 ;
                                00DC   260 ; CALLING SEQUENCE:
                                00DC   261 ;
                                00DC   262 ;       BSBW    MAP$IOSPACE
                                00DC   263 ;
                                00DC   264 ; INPUT PARAMETERS:     NONE
                                C0DC   265 ;
                                00DC   266 ;       R5      Address of P1 page tables
                                00DC   267 ;
                                00DC   268 ; IMPLICIT INPUTS:      NONE
                                00DC   269 ;
                                00DC   270 ; OUTPUT PARAMETERS:    NONE
                                00DC   271 ;
                                00DC   272 ; IMPLICIT OUTPUTS:
                                00DC   273 ;
                                00DC   274 ;       Updates P1 page tables
                                00DC   275 ;
                                00DC   276 ; SIDE EFFECTS:         NONE
                                00DC   277 ;--
```

```
                                        00DC   279 MAP$IOSPACE::
                                 18   BB 00DC   280          PUSHR   #^M<R3,R4>                 ; Save
               54    00F00000    8F   D0 00DE   281          MOVL    #^XF00000,R4              ; Lowest slot address
                                 7E   DF 00E5   282 10$:     PUSHAL  -(SP)                     ; Address to store longword
                                 10   BB 00E7   283          PUSHR   #^MR4                     ; Address to fetch
               0000012A'EF       02   FB 00E9   284          CALLS   #2,FETCH_LONG             ; Check it out
                                 08   BA 00F0   285          POPR    #^MR3                     ; Get contents
                              12 50   E9 00F2   286          BLBC    R0,50$                    ; Branch if nothing there
                                        00F5   287
                51    54    F7 8F   78 00F5   288          ASHL    #-9,R4,R1                 ; PFN of slot
               52    A0000000 8F   D0 00FA   289          MOVL    #PTE$M_VALID!PTE$C_UW,R2; PROTECTION
                        50   10   D0 0101   290          MOVL    #16,R0                    ; Map 16 pages
                           FEF9'  30 0104   291          BSBW    MAPMEM$IOSPACE            ; Call to setup PTE's
                                        0107   292
54    00002000 8F    00F7FFFF 8F   F1 C107   293 50$:     ACBL    #^XF7FFFF,#^X2000,R4,10$; Do each slot 0-63
                                    FFD0 0113
                                        0115   294
                   51    7FF0 8F   3C 0115   295          MOVZWL  #^XFFE000@-9,R1           ; PFN of unibus ^0760000
               52    A0000000 8F   D0 011A   296          MOVL    #PTE$M_VALID!PTE$C_UW,R2; PROTECTION
                        50   10   D0 0121   297          MOVL    #16,R0                    ; Map 16 pages
                           FED9'  30 0124   298          BSBW    MAPMEM$IOSPACE            ; Call to setup PTE's
                                        0127   299
                                 18   BA 0127   300          POPR    #^M<R3,R4>
                                      05 0129   301          RSB
```

```
                                012A   303          .SBTTL  FETCH_LONG      Retreive longword from possibly bad address
                                012A   304 ;++
                                012A   305 ; FUNCTIONAL DESCRIPTION:
                                012A   306 ;
                                012A   307 ;       This routine is used to fetch a longword from the designated location
                                012A   308 ;       It handles machine checks if they occur and return the indicated status
                                012A   309 ;
                                012A   310 ; CALLING SEQUENCE:
                                012A   311 ;
                                012A   312 ;       FETCH_LONG(ADDRESS,RESULT)
                                012A   313 ;
                                012A   314 ; INPUT PARAMETERS:
                                012A   315 ;
                                012A   316 ;       ADDRESS Address of Longword to fetch
                                C12A   317 ;
                                012A   318 ; IMPLICIT INPUTS:      NONE
                                012A   319 ;
                                012A   320 ; OUTPUT PARAMETERS:
                                012A   321 ;
                                012A   322 ;       RESULT  Address to store the fetched longword
                                012A   323 ;
                                012A   324 ; IMPLICIT OUTPUTS:     NONE
                                012A   325 ;
                                012A   326 ;--
                                012A   327
                          0000  012A   328 .ENTRY  FETCH_LONG,^M<>
                                012C   329
      6D     39'AF     9E      012C   330          MOVAB   B^20$,(FP)              ; Set exception handler
08 BC     04 BC     D0      0130   331          MOVL    @4(AP),@8(AP)           ; Fetch
         50    01     D0      0135   332          MOVL    S^#SS$_NORMAL,R0        ; Success
                      04      0138   333          RET
                              0139   334
                          0000  0139   335 20$:    .WORD   ^M<>                    ; Save no registers
            50    04 AC     7D      0138   336          MOVQ    4(AP),R0        ; Get signal and mechanism pointers
04 A0    00660088 8F     D1      013F   337          CMPL    #DS$_MCHK,4(R0)         ; Machine check?
                      0F      12      0147   338          BNEQ    30$             ; Branch if not
         CC A1    04 A0     D0      0149   339          MOVL    4(R0),12(R1)    ; Change return code
                      7E      7C      014E   340          CLRQ    -(SP)           ; Null args to unwind
   00000000'EF    02      FB      0150   341          CALLS   #2,SYS$UNWIND   ; Unwind the stack
                      04      0157   342          RET                     ; And return
                              0158   343
         50    0000'8F     3C      0158   344 30$:    MOVZWL  #SS$_RESIGNAL,R0        ; Let someone else have it
                      04      015D   345          RET                     ; Return
                              015E   346
```

```
                     015E   348          .SBTTL  PURGE DATAPATH
                     015E   349 ;+
                     015E   350 ; IOC$PURGDATAP - PURGE DATAPATH
                     015E   351 ;
                     015E   352 ; THIS ROUTINE PURGES THE CALLER'S BUFFERED DATAPATH, AND CLEARS ANY
                     015E   353 ; DATAPATH ERRORS.  IF THERE WAS A DATAPATH ERROR, THIS FACT IS
                     015E   354 ; RETURNED TO THE CALLER.
                     015E   355 ;
                     015E   356 ; INPUTS:
                     015E   357 ;
                     015E   358 ;       R5 = UCB ADDRESS
                     015E   359 ;
                     015E   360 ; OUTPUTS:
                     015E   361 ;
                     015E   362 ;       R0-R3 ALTERED
                     015E   363 ;       OTHER REGISTERS PRESERVED
                     015E   364 ;       R0 LOW BIT CLEAR/SET IF TRANSMISSION ERROR/SUCCESS
                     015E   365 ;       R1 = DPR CONTENTS AFTER PURGE (FOR REGISTER DUMP BY CALLER)
                     015E   366 ;       R2 = ADDRESS OF START OF ADAPTER MAP REGISTERS (FOR REG DUMP BY CALLER)
                     015E   367 ;       R3 = CRB ADDRESS
                     015E   368 ;-
                     015E   369
                     015E   370
                     015E   371
                     015E   372 IOC$PURGDATAP::
          53   20 A5  D0  015E   373          MOVL    UCB$L_CRB(R5),R3        ;CRB ADDRESS
          52   28 B3  D0  0162   374          MOVL    @CRB$L_INTD+VEC$L_ADP(R3),R2 ;GET START OF ADAPTER
    52  00000800 8F  C0  0166   375          ADDL2   #^X800, R2              ; Offset address to MAP regs.          [04]
        50   01  7D  016D   376          MOVQ    #1,R0                   ;R0=1, R1= 0
                     05  0170   377          RSB                             ;RETURN
```

```
                        0171   379 ;++
                        0171   380 ; 11/750 SPECIFIC UBI INITIALIZATION
                        0171   381 ; AND ADAPTER BLOCK BUILD
                        0171   382 ;--
                        0171   383 UBADP_CPU::
00000200'EF   DE        0171   384         MOVAL   L^SCB_BASE+^X200,-      ; UB0                                  [06]
          10 A2         0177   385                 ADP$L_VECTOR(R2)        ; VECTOR SPACE
 24 A2    0E   B0      0179   386         MOVW    #^XE,ADP$W_DPBITMAP(R2) ; MARK DATAPATHS 1-3 AVAILABLE
                  05    017D   387         RSB                             ; RETURN TO COMMON CODE
                        017E   388
                00000084 017E   389 UBINTSZ==^XB4                           ; Used by QIO                          [12]
```

```
                     017E    391          .SBTTL  UBA$INITIAL     UNIBUS ADAPTER INITIALIZATION
                     017E    392 ;+
                     017E    393 ; UBA$INITIAL - UNIBUS ADAPTER INITIALIZATION
                     017E    394 ;
                     017E    395 ; THIS ROUTINE IS CALLED VIA A JSB INSTRUCTION AT SYSTEM STARTUP AND AFTER
                     017E    396 ; A POWER RECOVERY RESTART TO ALLOW INITIALIZATION OF UNIBUS ADAPTERS.
                     017E    397 ;
                     017E    398 ; INPUTS:
                     017E    399 ;
                     017E    400 ;       R4 = ADDRESS OF UNIBUS ADAPTER CONFIGURATION STATUS REGISTER.
                     017E    401 ;
                     017E    402 ;       ALL INTERRUPTS ARE LOCKED OUT.
                     017E    403 ;
                     017E    404 ; OUTPUTS:
                     017E    405 ;
                     017E    406 ;       THE UNIBUS ADAPTER IS INITIALIZED AND INTERRUPTS ARE ENABLED.
                     017E    407 ;-
                     017E    408
                     017E    409 UBA$INITIAL::                               ;UNIBUS ADAPTER INITIALIZATION
                  05 017E    410          RSB                                ;
                     017F    411 ;
                     017F    412 ; IGNORE UNEXPECTED UNIBUS INTERRUPTS
                     017F    413 ;
                     017F    414
                     017F    415          .ALIGN  LONG
                     0180    416 UBA$UNEXINT::                               ; UNEXPECTED INTERRUPT CODE
                  02 0180    417          REI                                ; AND RETURN
```

```
                        0181   419          .SBTTL  IO$TESTCSR_x    Test CSR for existence
                        0181   420 ;++
                        0181   421 ;
                        0181   422 ; IO$TESTCSR_L - TEST A LONGWORD FOR EXISTENCE
                        0181   423 ; IO$TESTCSR_W - TEST A WORD FOR EXISTENCE IN I/O SPACE
                        0181   424 ;
                        0181   425 ; THIS TEST IS CPU DEPENDENT. THE FOLLOWING CPU'S ARE SUPPORTED:
                        0181   426 ;
                        0181   427 ;       11/780 - TEST CSR AND CHECK RESULT IN THE UBA STATUS REGISTER.
                        0181   428 ;       11/750 - NOM-EXISTENT CSR IS REPORTED VIA MACHINE CHECK AS A
                        0181   429 ;     & 11/730   NON-EXISTENT MEMORY REFERENCE. CONNECT A TEMPORARY
                        0181   430 ;                MACHINE CHECK HANDLER, TEST THE CSR, AND RESTORE THE
                        0181   431 ;                ORIGINAL MACHINE CHECK HANDLER.
                        0181   432 ;
                        0181   433 ; INPUTS:
                        0181   434 ;
                        0181   435 ;       R0 = CSR ADDRESS
                        0181   436 ;       R6 = ADAPTER CONFIGURATION REGISTER
                        0181   437 ;
                        0181   438 ; OUTPUTS:
                        0181   439 ;
                        0181   440 ;       R0 = LOW BIT SET/CLEAR FOR EXISTENT/NONEX CSR
                        0181   441 ;            OTHER REGISTERS ARE PRESERVED.
                        0181   442 ;
                        0181   443 ;--
                        0181   444 IO$TESTCSR_L::
                  02    BB    0181   445          PUSHR   #^M<R1>                 ; Save a register
      51  00000004'EF     DE  0183   446          MOVAL   SCB_BASE+4,R1           ; Base SCB
                  61    DD    018A   447          PUSHL   (R1)                    ; Save previous machine check handler
      61  00000000'EF     DE  018C   448          MOVAL   TST$MCHK,(R1)           ; Set machine check handler
                  60    D5    0193   449          TSTL    (R0)                    ; Test for existence
                  61  8ED0    0195   450          POPL    (R1)                    ; Restore previous MCHK handler
                  50    DC    0198   451          MOVPSL  R0                      ; C-bit set indicates MCHK
            50    50    D2    019A   452          MCOML   R0,R0                   ; C-bit clear indicates ok
                  02    BA    019D   453          POPR    #^M<R1>                 ; Restore
                        05    019F   454          RSB                             ; Return
                              01A0   455
                              01A0   456 IO$TESTCSR_W::
                  02    BB    01A0   457          PUSHR   #^M<R1>                 ; Save a register
      51  00000004'EF     DE  01A2   458          MOVAL   SCB_BASE+4,R1           ; Base SCB
                  61    DD    01A9   459          PUSHL   (R1)                    ; Save previous machine check handler
      61  00000000'EF     DE  01AB   460          MOVAL   TST$MCHK,(R1)           ; Set machine check handler
                  60    B5    01B2   461          TSTW    (R0)                    ; Test for existence
                  61  8ED0    01B4   462          POPL    (R1)                    ; Restore previous MCHK handler
                  50    DC    01B7   463          MOVPSL  R0                      ; C-bit set indicates MCHK
            50    50    D2    01B9   464          MCOML   R0,R0                   ; C-bit clear indicates ok
                  02    BA    01BC   465          POPR    #^M<R1>                 ; Restore
                        05    01BE   466          RSB                             ; Return
```

```
                                01BF    468          .SBTTL  IOGEN$CONN_VEC
                                01BF    469 ;++
                                01BF    470 ;
                                01BF    471 ; IOGEN$CONN_VEC - CONNECT A UNIBUS INTERRUPT DISPATCHER TO A VECTOR
                                01BF    472 ;
                                01BF    473 ; THIS SUBROUTINE IS CPU-DEPENDENT. THE FOLLOWING CPU'S ARE SUPPORTED:
                                01BF    474 ;
                                01BF    475 ;       11/780 - CONNECT VEC$Q_DISPATCH+2 (JSB @#) TO VECTOR
                                01BF    476 ;       11/750 - CONNECT VEC$Q_DISPATCH (PUSHR) TO VECTOR
                                01BF    477 ;
                                01BF    478 ; FOR ALL CPU'S, PUSHR #^M<R2,R3,R4,R5> IN THE INTERRUPT DISPATCH BLOCK
                                01BF    479 ; IS CHANGED TO PUSHR #^M<R0,R1,R2,R3,R4,R5>.
                                01BF    480 ;
                                01BF    481 ; INPUTS:
                                01BF    482 ;
                                01BF    483 ;       R0 = ADDRESS OF VECTOR TO CONNECT
                                01BF    484 ;       R4 = ADDRESS OF INTERRUPT DISPATCH BLOCK IN CRB
                                01BF    485 ;
                                01BF    486 ; OUTPUTS:
                                01BF    487 ;
                                01BF    488 ;       ALL REGISTERS PRESERVED
                                01BF    489 ;--
                                01BF    490 IOGEN$CONN_VEC::
        64  00000000'EF   B0    01BF    491          MOVW    UBINT_DISP,VEC$Q_DISPATCH(R4)   ; CHANGE PUSHR TO PUSH R0-R5
              60    64    9E    01C6    492          MOVAB   VEC$Q_DISPATCH(R4),(R0)         ; CONNECT PUSHR TO VECTOR
                          05    01C9    493          RSB                                     ; RETURN
```

```
                                01CA   495          .SBTTL  QIOCLEAN_CPU
                                01CA   496 ;
                                01CA   497 ; CPU SPECIFIC QIO CLEANUP
                                01CA   498 ;       R4 = SCB_IMAGE
                                01CA   499 ;       R5 = SCB_BASE
                                01CA   500 ;
                                01CA   501 QIOCLEAN_CPU::
                      33    BB  01CA   502          PUSHR   #^M<R0,R1,R4,R5>        ; SAVE R0,R1,R4,R5
              50    0080 8F   3C  01CC   503          MOVZWL  #<512/4>,R0             ; Start with SCB page 2, offset ^X200
    6540   00000000'EF40   DE  01D1   504 10$:      MOVAL   SCB_UNKINT[R0],(R5)[R0] ; Re-fill SCB with pointer to UNK_INT
     EF 50    00000180 8F   F2  01DA   505          AOBLSS  #3*<512/4>,R0,10$       ; Fill 2 SCB vector pages
            53    0000'CF   DE  01E2   506          MOVAL   W^IOC$GL_ADPLIST,R3     ; POINT TO THE ADAPTER HEADER
                                01E7   507 20$:
                50    63   D0  01E7   508          MOVL    (R3),R0                 ; POINT TO THE FIRST ADAPTER BLOCK
                      09   13  01EA   509          BEQL    30$                     ; FINISH IF EMPTY
                63    04 A0   D0  01EC   510          MOVL    ADP$L_LINK(R0),(R3)     ; LINK TO THE NEXT
                        FEOD'  30  01F0   511          BSBW    QIO$DEALLOCATE          ; RELEASE THE ADAPTER CONTROL BLOCK
                          F2   11  01F3   512          BRB     20$                     ; RELEASE THEM ALL
                                01F5   513 30$:
                      33    BA  01F5   514          POPR    #^M<R0,R1,R4,R5>        ; RESTORE R0,R1,R4,R5
                            05  01F7   515          RSB                             ; Return
```

```
                        01F8    517          .SBTTL  ONLY_SCB        CPU specific SCB setup
                        01F8    518 ;++                                                          [08]
                        01F8    519 ; FUNCTIONAL DESCRIPTION:
                        01F8    520 ;
                        01F8    521 ;       Does nothing for the case of NEBULA.
                        01F8    522 ;
                        01F8    523 ;--                                                          [08]
                        01F8    524                                                              [08]
                        01F8    525 ONLY_SCB::                          ;                       [08]
                    05  01F8    526          RSB                        ;                       [08]
```

```
                        01F9   528          .SBTTL  SCBVECTOR_xxx   Otherwise undefined SCB vectors
                        01F9   529 ;++
                        01F9   530 ; FUNCTIONAL DESCRIPTION:
                        01F9   531 ;
                        01F9   532 ;       These interrupt entry points declare unexpected exceptions
                        01F9   533 ;       for all unused SCB vectors.
                        01F9   534 ;
                        01F9   535 ;--
                        01F9   536          .LIST   MEB,MC
                        01F9   537
                        01F9   538          .MACRO  SPAREVECTOR     OFFSET
                        01F9   539          .ALIGN  LONG
                        01F9   540 SCBVECTOR_'OFFSET'==.+1S
                        01F9   541          BSBB    COMMON_SCB                      ; Enter common handler
                        01F9   542          .WORD   ^X'OFFSET'
                        01F9   543          .ENDM
                        01F9   544
                        01F9   545          SPAREVECTOR     000
                        01F9               .ALIGN  LONG
                7A   10 01FC               BSBB    COMMON_SCB                      ; Enter common handler
                  0000  01FE               .WORD   ^X000
                        0200   546          SPAREVECTOR     038
                76   10 0200               BSBB    COMMON_SCB                      ; Enter common handler
                  0038  0202               .WORD   ^X038
                        0204   547          SPAREVECTOR     03C
                72   10 0204               BSBB    COMMON_SCB                      ; Enter common handler
                  003C  0206               .WORD   ^X03C
                        020R   548          SPAREVECTOR     050
                6E   10 0208               BSBB    COMMON_SCB                      ; Enter common handler
                  0050  020A               .WORD   ^X050
                        020C   549          SPAREVECTOR     054
                6A   10 020C               BSBB    COMMON_SCB                      ; Enter common handler
                  0054  020E               .WORD   ^X054
                        0210   550          SPAREVECTOR     058
                66   10 0210               BSBB    COMMON_SCB                      ; Enter common handler
                  0058  0212               .WORD   ^X058
                        0214   551          SPAREVECTOR     05C
                62   10 0214               BSBB    COMMON_SCB                      ; Enter common handler
                  005C  0216               .WORD   ^X05C
                        0218   552          SPAREVECTOR     060
                5E   10 0218               BSBB    COMMON_SCB                      ; Enter common handler
                  0060  021A               .WORD   ^X060
                        021C   553          SPAREVECTOR     064
                5A   10 021C               BSBB    COMMON_SCB                      ; Enter common handler
                  0064  021E               .WORD   ^X064
                        0220   554          SPAREVECTOR     068
                56   10 0220               BSBB    COMMON_SCB                      ; Enter common handler
                  0068  0222               .WORD   ^X068
                        0224   555          SPAREVECTOR     06C
                52   10 0224               BSBB    COMMON_SCB                      ; Enter common handler
                  006C  0226               .WORD   ^X06C
                        0228   556          SPAREVECTOR     070
                4E   10 0228               BSBB    COMMON_SCB                      ; Enter common handler
                  0070  022A               .WORD   ^X070
                        022C   557          SPAREVECTOR     074
                4A   10 022C               BSBB    COMMON_SCB                      ; Enter common handler
                  0074  022E               .WORD   ^X074
```

```
                     0230   558          SPAREVECTOR     078
                46   10 0230            BSBB    COMMON_SCB                      ; Enter common handler
                   0078 0232            .WORD   ^X078
                     0234   559          SPAREVECTOR     07C
                42   10 0234            BSBB    COMMON_SCB                      ; Enter common handler
                   007C 0236            .WORD   ^X07C
                     0238   560          SPAREVECTOR     080
                3E   10 0238            BSBB    COMMON_SCB                      ; Enter common handler
                   0080 023A            .WORD   ^X080
                     023C   561          SPAREVECTOR     0C4
                3A   10 023C            BSBB    COMMON_SCB                      ; Enter common handler
                   00C4 023E            .WORD   ^X0C4
                     0240   562          SPAREVECTOR     0C8
                36   10 0240            BSBB    COMMON_SCB                      ; Enter common handler
                   00C8 C242            .WORD   ^X0C8
                     0244   563          SPAREVECTOR     0CC
                32   10 0244            BSBB    COMMON_SCB                      ; Enter common handler
                   00CC 0246            .WORD   ^X0CC
                     0248   564          SPAREVECTOR     0D0
                2E   10 0248            BSBB    COMMON_SCB                      ; Enter common handler
                   00D0 024A            .WORD   ^X0D0
                     024C   565          SPAREVECTOR     0D4
                2A   10 024C            BSBB    COMMON_SCB                      ; Enter common handler
                   00D4 024E            .WORD   ^X0D4
                     0250   566          SPAREVECTOR     0D8
                26   10 0250            BSBB    COMMON_SCB                      ; Enter common handler
                   00D8 0252            .WORD   ^X0D8
                     0254   567          SPAREVECTOR     0DC
                22   10 0254            BSBB    COMMON_SCB                      ; Enter common handler
                   00DC 0256            .WORD   ^X0DC
                     0258   568          SPAREVECTOR     0E0
                1E   10 0258            BSBB    COMMON_SCB                      ; Enter common handler
                   00E0 025A            .WORD   ^X0E0
                     025C   569          SPAREVECTOR     0E4
                1A   10 025C            BSBB    COMMON_SCB                      ; Enter common handler
                   00E4 025E            .WORD   ^X0E4
                     0260   570          SPAREVECTOR     0E8
                16   10 0260            BSBB    COMMON_SCB                      ; Enter common handler
                   00E8 0262            .WORD   ^X0E8
                     0264   571          SPAREVECTOR     0EC
                12   10 0264            BSBB    COMMON_SCB                      ; Enter common handler
                   00EC 0266            .WORD   ^X0EC
                     0268   572          SPAREVECTOR     0F0
                0E   10 0268            BSBB    COMMON_SCB                      ; Enter common handler
                   00F0 026A            .WORD   ^X0F0
                     026C   573          SPAREVECTOR     0F4
                0A   10 026C            BSBB    COMMON_SCB                      ; Enter common handler
                   00F4 026E            .WORD   ^X0F4
                     0270   574          SPAREVECTOR     0F8
                06   10 0270            BSBB    COMMON_SCB                      ; Enter common handler
                   00F8 0272            .WORD   ^X0F8
                     0274   575          SPAREVECTOR     0FC
                02   10 0274            BSBB    COMMON_SCB                      ; Enter common handler
                   00FC 0276            .WORD   ^X0FC
                     0278   576 COMMON_SCB:
            6E 00 BE   3C 0278   577          MOVZWL  @(SP),(SP)              ; Get vector offset
                FD81'  31 027C   578          BRW     EXE_UNEXPINT            ; Generate unexpected interrupt
```

```
                        580          .SBTTL  ONLY_CLOCK_INIT Perform cpu-specific TODR initialization
                        581 ;++                                                                    begin   [11]
                        582 ; FUNCTIONAL DESCRIPTION:
                        583 ;
                        584 ;       This routine only reads the TODR with an MFPR.
                        585 ;
                        586 ; CALLING SEQUENCE:
                        587 ;
                        588 ;       JSB     ONLY_CLOCK_INIT
                        589 ;
                        590 ; INPUT PARAMETERS:
                        591 ;
                        592 ;       None
                        593 ;
                        594 ; IMPLICIT INPUTS:
                        595 ;
                        596 ;       None
                        597 ;
                        598 ; OUTPUT PARAMETERS:
                        599 ;
                        600 ;       R0      = Contents of TODR
                        601 ;
                        602 ; IMPLICIT OUTPUTS:
                        603 ;
                        604 ;       None
                        605 ;
                        606 ; COMPLETION CODES:
                        607 ;
                        608 ;       None
                        609 ;
                        610 ; SIDE EFFECTS:
                        611 ;
                        612 ;       None
                        613 ;
                        614 ; REGISTER USAGE:
                        615 ;
                        616 ;       R0      Contents of TODR
                        617 ;--
```

```
                          027F    619 ONLY_CLOCK_INIT::                           ;                                       [11]
                    50    1B  DB 027F    620          MFPR    #PR730$_TODR,R0    ; Fetch the TODR                        [11]
                          05  0282    621          RSB                            ;                                       [11]
                              0283    622          .END
```

```
$ER                =  00000002     D
$MODULE              00000000 R  D  03
ADP$B_NUMBER         0000000B    D
ADP$B_PORT           00000020    D
ADP$B_TYPE           0000000A    D
ADP$C_DRADPLEN       00000014    D
ADP$C_MBAADPLEN      00000014    D
ADP$C_MPMADPLEN      00000070    D
ADP$C_UBAADPLEN      00000070    D
ADP$K_DRADPLEN       00000014    D
ADP$K_MBAADPLEN      00000014    D
ADP$K_MPMADPLEN      00000070    D
ADP$K_UBAADPLEN      00000070    D
ADP$L_CRB            00000010    D
ADP$L_CSR            00000000    D
ADP$L_DPQBL          00000018    D
ADP$L_DPQFL          00000014    D
ADP$L_INTD           00000064    D
ADP$L_LINK           00000004    D
ADP$L_MRQBL          00000020    D
ADP$L_MRQFL          0000001C    D
ADP$L_PRQQBL         00000018    D
ADP$L_PRQQFL         00000014    D
ADP$L_SHB            0000001C    D
ADP$L_VECTOR         00000010    D
ADP$W_ADPTYPE        0000000E    D
ADP$W_DPBITMAP       00000024    D
ADP$W_MRBITMAP       00000026    D
ADP$W_SIZE           00000008    D
ADP$W_TR             0000000C    D
BIT...             =  00660130    D
BUG$CHECK            ********  X    00
CMK$RCONSOLE         ********  X    00
CMK$TCONSOLE         ********  X    00
COMMON_SCB           00000278 R    D  04
CONFIG_EXIT          000000AA R    D  04
CONFIG_RELEASE       0000009D R    D  04
CRB$B_MASK           0000000E      D
CRB$B_TYPE           0000000A      D
CRB$C_LENGTH         00000038      D
CRB$K_LENGTH         00000038      D
CRB$L_INTD           00000014      D
CRB$L_INTD2          00000038      D
CRB$L_LINK           00000010      D
CRB$L_WQBL           00000004      D
CRB$L_WQFL           00000000      D
CRB$W_REFC           0000000C      D
CRB$W_SIZE           00000008      D
DIR...             =  FFFFFFFF     D
DS$GK_SID          =  03000000  G  D
DS$GK_USOVR        =  00000200  G  D
DS$K_ERROR         =  00000002     D
DS$K_NORMAL        =  00000001     D
DS$K_SEVERE        =  00000004     D
DS$K_SUBSYS        =  00000066     D
DS$K_WARNING       =  00000000     D
DS$WAITUS            ********  X    00
DS$_ARITH          =  006600D0     D
DS$_ASBE           =  00660118     D
DS$_BADLINK        =  006600F0     D
DS$_BADTYPE        =  006600E8     D
DS$_BIIC           =  00660120     D
DS$_CHME           =  006600A8     D
DS$_CHMK           =  006600E0     D
DS$_DEVNAME        =  00660108     D
DS$_ERROR          =  00660002     D
DS$_FHWE           =  00660068     D
DS$_FRAGBUF        =  00660080     D
DS$_ICBUSY         =  006600C8     D
DS$_ICERR          =  006600C0     D
DS$_IHWE           =  00660060     D
DS$_ILLCHAR        =  00660018     D
DS$_ILLPAGCNT      =  00660078     D
DS$_ILLUNIT        =  00660100     D
DS$_INSFMEM        =  00660050     D
DS$_IPL2HI         =  006600B8     D
DS$_IVADDR         =  00660040     D
DS$_IVVECT         =  00660038     D
DS$_KRNLSTK        =  00660090     D
DS$_LOGIC          =  00660070     D
DS$_MCHK           =  00660088     D
DS$_MMOFF          =  00660058     D
DS$_NEEDUNIT       =  006600F8     D
DS$_NODE           =  00660128     D
DS$_NOPCS          =  00660110     D
DS$_NORMAL         =  00660001     D
DS$_NOSUPPORT      =  006600B1     D
DS$_NOTDON         =  00660030     D
DS$_NOTIMP         =  006600B0     D
DS$_NULLSTR        =  00660010     D
DS$_OVERFLOW       =  00660008     D
DS$_POWER          =  00660098     D
DS$_PROGERR        =  00660020     D
DS$_SEVERE         =  00660004     D
DS$_TRANSL         =  006600A0     D
DS$_TRUNCATE       =  00660028     D
DS$_UNEXPINT       =  006600D8     D
DS$_VASFULL        =  00660048     D
DS$_WARNING        =  00660000     D
DSP$CONFIG_ADP       00000000 RG   D  04
DS_ERRSUP            ********    X    00
DW730$K_LEN          00000032      D
EXE$ALONONPAGED      ********    X    00
EXE$DEANONPAGED      ********    X    00
EXE_UNEXPINT         ********    X    00
FETCH_LONG           0000012A RG   D  04
FETCH_STORE          ********    X    00
HANDLR               000000AB R    D  04
HP$A_DEPENDENT       00000032      D
HP$A_DEVICE          00000018      D
HP$A_DVA             0000001C      D
HP$A_LINK            00000020      D
HP$B_DRIVE           00000008      D
HP$B_FLAGS           0000000A      D
HP$Q_DEVICE          00000000      D
HP$T_DEVICE          0000000C      D
HP$T_TYPE            00000026      D
HP$W_SIZE            00000008      D
HP$W_VECTOR          00000024      D
INS$PTABLE           ********    X    00
IO$GQ_PHYSICAL       00000008 RG   D  03
IO$TESTCSR_L         00000181 RG   D  04
IO$TESTCSR_W         000001A0 RG   D  04
IOC$GL_ADPLIST       ********    X    00
IOC$K_IOSPACE      =  40000000  G  D
IOC$PURGDATAP        0000015E RG   D  04
IOGEN$CONN_VEC       000001BF RG   D  04
IS                 =  00000001     D
LOCAL_STORE          00000000 R    D  02
L_CONFIG             FFFFFFF4      D
L_IPL                FFFFFFF8      D
L_LOCAL              FFFFFFF4      D
L_VECTOR             FFFFFFFC      D
MAP$IOSPACE          000000DC RG   D  04
MAPMEM$IOSPACE       ********    X    00
ONLY_CLOCK_INIT      0000027F RG   D  04
ONLY_SCB             000001F8 RG   D  04
PR$_IPL            =  00000012     D
PR730$_TODR        =  0000001B     D
PTE$C_UW           =  20000000     D
PTE$M_IO           =  00100000     D
PTE$M_VALID        =  80000000     D
PTE$V_IO           =  00000014     D
QIO$DEALLOCATE       ********    X    00
QIOCLEAN_CPU         000001CA RG   D  04
SAVABS...          =  FFFFFFF4     D
SCBVECTOR_000      =  000001FD RG  D  04
SCBVECTOR_038      =  00000201 RG  D  04
SCBVECTOR_03C      =  00000205 RG  D  04
SCBVECTOR_050      =  00000209 RG  D  04
SCBVECTOR_054      =  0000020D RG  D  04
SCBVECTOR_058      =  00000211 RG  D  04
SCBVECTOR_05C      =  00000215 RG  D  04
SCBVECTOR_060      =  00000219 RG  D  04
SCBVECTOR_064      =  0000021D RG  D  04
SCBVECTOR_068      =  00000221 RG  D  04
SCBVECTOR_06C      =  00000225 RG  D  04
SCBVECTOR_070      =  00000229 RG  D  04
SCBVECTOR_074      =  0000022D RG  D  04
SCBVECTOR_078      =  00000231 RG  D  04
SCBVECTOR_07C      =  00000235 RG  D  04
SCBVECTOR_080      =  00000239 RG  D  04
SCBVECTOR_0C4      =  0000023D RG  D  04
SCBVECTOR_0C8      =  00000241 RG  D  04
SCBVECTOR_0CC      =  00000245 RG  D  04
SCBVECTOR_0D0      =  00000249 RG  D  04
SCBVECTOR_0D4      =  0000024D RG  D  04
SCBVECTOR_0D8      =  00000251 RG  D  04
SCBVECTOR_0DC      =  00000255 RG  D  04
SCBVECTOR_0E0      =  00000259 RG  D  04
SCBVECTOR_0E4      =  0000025D RG  D  04
```

```
SCBVECTOR_0E8  = 00000261 RG D  04      UCB$K_LENGTH      00000074  D      UCB$W_CYLINDERS   0000003E  D
SCBVECTOR_0EC  = 00000265 RG D  04      UCB$K_MB_LENGTH   00000090  D      UCB$W_DA          0000008C  D
SCBVECTOR_0F0  = 00000269 RG D  04      UCB$K_TT_LENGTH   000000BC  D      UCB$W_DC          0000008E  D
SCBVECTOR_0F4  = 0000026D RG D  04      UCB$L_AMB         00000054  D      UCB$W_DEVBUFSIZ   0000003A  D
SCBVECTOR_0F8  = 00000271 RG D  04      UCB$L_ASTQBL      00000010  D      UCB$W_DEVSTS      0000005A  D
SCBVECTOR_0FC  = 00000275 RG D  04      UCB$L_ASTQFL      0000000C  D      UCB$W_DIRSEQ      00000088  D
SCB_BASE         ********    X  00      UCB$L_CPID        0000005C  D      UCB$W_DSTADDR     00000018  D
SCB_IMAGE        ********    X  00      UCB$L_CRB         00000020  D      UCB$W_DX_BCR      000000A4  D
SCB_UNIKINT      ********    X  00      UCB$L_DDB         00000024  D      UCB$W_ECT         00000090  D
SIZ...         = 00000001      D        UCB$L_DEVCHAR     00000034  D      UCB$W_EC2         00000092  D
SS$_CONTINUE     ********    X  00      UCB$L_DEVDEPEND   0000003C  D      UCB$W_ERRCNT      00000072  D
SS$_CONTROLC     ********    X  00      UCB$L_DPC         00000080  D      UCB$W_FUNC        0000007E  D
SS$_CTRLERR      ********    X  00      UCB$L_DUETIM      0000005C  D      UCB$W_MB_SEED     00000000  D
SS$_DATACHECK    ********    X  00      UCB$L_DX_BFPNT    0000009C  D      UCB$W_MSGCNT      00000016  D
SS$_NORMAL     = 00000001      D        UCB$L_DX_BUF      00000098  D      UCB$W_MSGMAX      00000014  D
SS$_RESIGNAL     ********    X  00      UCB$L_DX_RXDB     000000A0  D      UCB$W_NT_CHAN     0000007C  D
SYS$SETPRT       ********    X  00      UCB$L_EMB         00000078  D      UCB$W_OFFSET      0000008A  D
SYS$UNWIND       ********    X  00      UCB$L_FIRST       00000014  D      UCB$W_REFC        00000050  D
TST$MCHK         ********    X  00      UCB$L_FPC         0000000C  D      UCB$W_SIZ         00000008  D
UBA$INITIAL      0000017E RG D  04      UCB$L_FQBL        00000004  D      UCB$W_SRCADDR     0000001A  D
UBA$UNEXINT      00000180 RG D  04      UCB$L_FQFL        00000000  D      UCB$W_STS         00000058  D
UBADP_CPU        00000171 RG D  04      UCB$L_FR3         00000010  D      UCB$W_TT_DESIZE   000000A5  D
UBINTSZ        = 000000B4  G D         UCB$L_FR4         00000014  D      UCB$W_UNIT        00000048  D
UBINT_DISP       ********    X  00      UCB$L_IOQBL       00000044  D      UCB$W_VPROT       0000001A  D
UCB$B_AMOD       00000053      D        UCB$L_IOQFL       00000040  D      VEC$B_DATAPATH    00000013  D
UCB$B_CEX        00000077      D        UCB$L_IRP         0000004C  D      VEC$B_NUMREG      00000012  D
UCB$B_CM1        0000004A      D        UCB$L_LINK        0000002C  D      VEC$C_LENGTH      00000024  D
UCB$B_CM2        0000004B      D        UCB$L_LOGADR      00000064  D      VEC$K_LENGTH      00000024  D
UCB$B_DEVCLASS   00000038      D        UCB$L_MAXBLOCK    00000084  D      VEC$L_ADP         00000014  D
UCB$B_DEVTYPE    00000039      D        UCB$L_MB_MBX      0000007C  D      VEC$L_IDB         00000008  D
UCB$B_DIPL       00000052      D        UCB$L_MB_PORT     0000008C  D      VEC$L_INITIAL     0000000C  D
UCB$B_DX_SCTCNT  000000A6      D        UCB$L_MB_RAST     00000078  D      VEC$L_START       0000001C  D
UCB$B_ERTCNT     00000070      D        UCB$L_MB_SHB      00000080  D      VEC$L_UNITDISC    00000020  D
UCB$B_ERTMAX     00000071      D        UCB$L_MB_WAST     00000074  D      VEC$L_UNITINIT    00000018  D
UCB$B_FEX        00000076      D        UCB$L_MB_WIOQBL   00000088  D      VEC$Q_DISPATCH    00000000  D
UCB$B_FIPL       00000008      D        UCB$L_MB_WIOQFL   00000084  D      VEC$W_MAPREG      00000010  D
UCB$B_LOCSRV     0000003C      D        UCB$L_MEDIA       0000008C  D
UCB$B_OFFNDX     00000094      D        UCB$L_NT_DATSSB   00000074  D
UCB$B_OFFRTC     00000095      D        UCB$L_NT_INTSSB   00000078  D
UCB$B_REMSRV     0000003D      D        UCB$L_OPCNT       00000060  D
UCB$B_SECTORS    0000003C      D        UCB$L_OWNUIC      0000001C  D
UCB$B_SLAVE      00000074      D        UCB$L_PID         00000028  D
UCB$B_SPR        00000075      D        UCB$L_RQBL        00000004  D
UCB$B_STATE      00000052      D        UCB$L_RQFL        00000000  D
UCB$B_TRACKS     0000003D      D        UCB$L_SVAPTE      00000068  D
UCB$B_TT_CRFILL  0000009D      D        UCB$L_SVPN        00000064  D
UCB$B_TT_DECRF   000000A1      D        UCB$L_TT_DECHAR   000000A8  D
UCB$B_TT_DELFF   000000A2      D        UCB$L_TT_RDUE     0000008C  D
UCB$B_TT_DESPEE  000000A0      D        UCB$L_TT_RTIMOU   000000B8  D
UCB$B_TT_DETYPE  000000A4      D        UCB$L_VCB         00000030  D
UCB$B_TT_LFFILL  0000009E      D        UCB$T_PARTNER     0000000C  D
UCB$B_TT_SPEED   0000009C      D        UCB$W_BCNT        0000006E  D
UCB$B_TYPE       0000000A      D        UCB$W_BCR         00000096  D
UCB$B_VERTSZ     0000003F      D        UCB$W_BOFF        0000006C  D
UCB$C_LENGTH     00000074      D        UCB$W_BUFQUO      00000018  D
UCB$C_MB_LENGTH  00000090      D        UCB$W_BYTESTOGO   0000003E  D
UCB$C_TT_LENGTH  000000BC      D        UCB$W_CHARGE      0000004A  D
```

Psect synopsis

PSECT name	Allocation		PSECT No.		Attributes									
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC BYTE
ABS	FFFFFFFC	(0.)	01	(1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
WORK	00000004	(4.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	NOVEC LONG
DATA	00000010	(16.)	03	(3.)	NOPIC	USR	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	NOVEC LONG
CODE	00000283	(643.)	04	(4.)	NOPIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	201 (3)	#-201 (3)
$MODULE	00000000-R	132 (1)	201 (3)
ADP$L_LINK	00000004		#-510 (13)
ADP$L_VECTOR	00000010		#-385 (9)
ADP$W_DPBITMAP	00000024		#-386 (9)
BIT...	=00660130	94 (1)	88 (1) 94 (1)
BUG$CHECK	00000000-XR		111 (1)
CMK$RCONSOLE	00000000-XR		111 (1)
CMK$TCONSOLE	00000000-XR		111 (1)
COMMON_SCB	00000278-R	576 (15)	#-545 (15) #-546 (15) #-547 (15) #-548 (15) #-549 (15)
			#-550 (15) #-551 (15) #-552 (15) #-553 (15) #-554 (15)
			#-555 (15) #-556 (15) #-557 (15) #-558 (15) #-559 (15)
			#-560 (15) #-561 (15) #-562 (15) #-563 (15) #-564 (15)
			#-565 (15) #-566 (15) #-567 (15) #-568 (15) #-569 (15)
			#-570 (15) #-571 (15) #-572 (15) #-573 (15) #-574 (15)
			#-575 (15)
CONFIG_EXIT	000000AA-R	233 (3)	#-227 (3)
CONFIG_RELEASE	0000009D-R	229 (3)	
CRB$L_INTD	00000014		#-374 (8)
DIR...	=FFFFFFFF	182 (2)	182 (2)
DS$GK_SID	=03000000	109 (1)	
DS$GK_USOVR	=00000200	103 (1)	
DS$K_ERROR	=00000002	94 (1)	
DS$K_NORMAL	=00000001	94 (1)	
DS$K_SEVERE	=00000004	94 (1)	
DS$K_SUBSYS	=00000066	94 (1)	94 (1)
DS$K_WARNING	=00000000	94 (1)	
DS$WAITUS	00000000-XR		111 (1)
DS$_ARITH	=006600D0	94 (1)	
DS$_ASBE	=00660118	94 (1)	
DS$_BADLINK	=006600F0	94 (1)	
DS$_BADTYPE	=006600E8	94 (1)	
DS$_BIIC	=00660120	94 (1)	
DS$_CHME	=006600A8	94 (1)	
DS$_CHMK	=006600E0	94 (1)	
DS$_DEVNAME	=00660108	94 (1)	
DS$_ERROR	=00660002	94 (1)	#-202 (3) #-232 (3)
DS$_FHWE	=00660068	94 (1)	
DS$_FRAGBUF	=00660080	94 (1)	
DS$_ICBUSY	=006600C8	94 (1)	
DS$_ICERR	=006600C0	94 (1)	
DS$_IHWE	=00660060	94 (1)	
DS$_ILLCHAR	=00660018	94 (1)	
DS$_ILLPAGCNT	=00660078	94 (1)	
DS$_ILLUNIT	=00660100	94 (1)	
DS$_INSFMEM	=00660050	94 (1)	
DS$_IPL2HI	=006600B8	94 (1)	
DS$_IVADDR	=00660040	94 (1)	
DS$_IVVECT	=00660038	94 (1)	
DS$_KRNLSTK	=00660090	94 (1)	

```
DS$_LOGIC          =00660070      94      (1)
DS$_MCHK           =00660088      94      (1)     #-337     (7)
DS$_MMOFF          =00660058      94      (1)
DS$_NEEDUNIT       =006600F8      94      (1)
DS$_NODE           =00660128      94      (1)
DS$_NOPCS          =00660110      94      (1)
DS$_NORMAL         =00660001      94      (1)
DS$_NOSUPPORT      =006600B1      94      (1)
DS$_NOTDON         =00660030      94      (1)
DS$_NOTIMP         =006600B0      94      (1)       94      (1)
DS$_NULLSTR        =00660010      94      (1)
DS$_OVERFLOW       =00660008      94      (1)
DS$_POWER          =00660098      94      (1)
DS$_PROGERR        =00660020      94      (1)
DS$_SEVERE         =00660004      94      (1)
DS$_TRANSL         =006600A0      94      (1)
DS$_TRUNCATE       =00660028      94      (1)
DS$_UNEXPINT       =006600D8      94      (1)     #-241     (4)
DS$_VASFULL        =00660048      94      (1)
DS$_WARNING        =00660000      94      (1)       94      (1)
DSP$CONFIG_ADP      00000000-R    188     (3)
DS_ERRSUP           00000000-XR                     112     (1)       201     (3)
DW730$K_LEN         00000032                      #-206     (3)
EXE$ALONONPAGED     00000000-XR                     114     (1)     #-207     (3)
EXE$DEANONPAGED     00000000-XR                     114     (1)     #-231     (3)
EXE_UNEXPINT        00000000-XR                     113     (1)     # 578     (15)
FETCH_LONG          0000012A-R    328     (7)       284     (6)
FETCH_STORE         00000000-XR                     114     (1)
HANDLR              000000AB-R    236     (4)
HP$A_DEVICE         00000018                      #-222     (3)
HP$A_DVA            0000001C                      #-212     (3)
HP$A_LINK           00000020                      #-224     (3)
HP$Q_DEVICE         00000000                      #-211     (3)     #-221     (3)
HP$T_DEVICE         0000000C                      #-214     (3)       220     (3)
HP$T_TYPE           00000026                      #-215     (3)     #-216     (3)
HP$W_SIZE           00000008                      #-213     (3)
HP$W_VECTOR         00000024                      #-217     (3)
INS$PTABLE          00000000-XR                     115     (1)       226     (3)
IO$GQ_PHYSICAL      00000008-R    133     (1)
IO$TESTCSR_L        00000181-R    444     (11)
IO$TESTCSR_W        000001A0-R    456     (11)
IOC$GL_ADPLIST      00000000-XR                     115     (1)       506     (13)
IOC$K_IOSPACE       =40000000     108     (1)     #-196     (3)
IOC$PURGDATAP       0000015E-R    372     (8)
IOGEN$CONN_VEC      000001BF-R    490     (12)
IS                  =00000001     98      (1)       545     (15)      546     (15)      547     (15)      548     (15)      549     (15)
                                                    550     (15)      551     (15)      552     (15)      553     (15)      554     (15)
                                                    555     (15)      556     (15)      557     (15)      558     (15)      559     (15)
                                                    560     (15)      561     (15)      562     (15)      563     (15)      564     (15)
                                                    565     (15)      566     (15)      567     (15)      568     (15)      569     (15)
                                                    570     (15)      571     (15)      572     (15)      573     (15)      574     (15)
                                                    575     (15)
LOCAL_STORE         00000000-R    128     (1)     #-192     (3)     #-237     (4)
L_CONFIG            FFFFFFF4      182     (2)     #-197     (3)     #-245     (4)
L_IPL               FFFFFFF8      182     (2)     #-194     (3)     #-244     (4)
L_LOCAL             FFFFFFF4      182     (2)       191     (3)
L_VECTOR            FFFFFFFC      182     (2)     #-193     (3)     #-243     (4)
```

```
MAP$IOSPACE      000000DC-R   279     (6)
MAPMEM$IOSPACE   00000000-XR                  116     (1)     #-291     (6)     #-298     (6)
ONLY_CLOCK_INIT  0000027F-R   619     (15)
ONLY_SCB         000001F8-R   525     (14)
PR$_IPL          =00000012                    #-244     (4)
PR730$_TODR      =0000001B                    #-620     (15)
PTE$C_UW         =20000000                    #-289     (6)     #-296     (6)
PTE$M_IO         =00100000    88      (1)
PTE$M_VALID      =80000000                    #-289     (6)     #-296     (6)
PTE$V_IO         =00000014    88      (1)
QIO$DEALLOCATE   00000000-XR                  117     (1)     #-511     (13)
QIOCLEAN_CPU     000001CA-R   501     (13)
SAVABS...        =FFFFFFF4    182     (2)
SCBVECTOR_000    =000001FD-R  545     (15)
SCBVECTOR_038    =00000201-R  546     (15)
SCBVECTOR_03C    =00000205-R  547     (15)
SCBVECTOR_050    =00000209-R  548     (15)
SCBVECTOR_054    =0000020D-R  549     (15)
SCBVECTOR_058    =00000211-R  550     (15)
SCBVECTOR_05C    =00000215-R  551     (15)
SCBVECTOR_060    =00000219-R  552     (15)
SCBVECTOR_064    =0000021D-R  553     (15)
SCBVECTOR_068    =00000221-R  554     (15)
SCBVECTOR_06C    =00000225-R  555     (15)
SCBVECTOR_070    =00000229-R  556     (15)
SCBVECTOR_074    =0000022D-R  557     (15)
SCBVECTOR_078    =00000231-R  558     (15)
SCBVECTOR_07C    =00000235-R  559     (15)
SCBVECTOR_080    =00000239-R  560     (15)
SCBVECTOR_0C4    =0000023D-R  561     (15)
SCBVECTOR_0C8    =00000241-R  562     (15)
SCBVECTOR_0CC    =00000245-R  563     (15)
SCBVECTOR_0D0    =00000249-R  564     (15)
SCBVECTOR_0D4    =0000024D-R  565     (15)
SCBVECTOR_0D8    =00000251-R  566     (15)
SCBVECTOR_0DC    =00000255-R  567     (15)
SCBVECTOR_0E0    =00000259-R  568     (15)
SCBVECTOR_0E4    =0000025D-R  569     (15)
SCBVECTOR_0E8    =00000261-R  570     (15)
SCBVECTOR_0EC    =00000265-R  571     (15)
SCBVECTOR_0F0    =00000269-R  572     (15)
SCBVECTOR_0F4    =0000026D-R  573     (15)
SCBVECTOR_0F8    =00000271-R  574     (15)
SCBVECTOR_0FC    =00000275-R  575     (15)
SCB_BASE         00000000-XR                  118     (1)       384     (9)       446     (11)      458     (11)
SCB_IMAGE        00000000-XR                  118     (1)
SCB_UNKINT       00000000-XR                  124     (1)       504     (13)
SIZ...           =00000001    88      (1)     88      (1)
SS$_CONTINUE     00000000-XR                  119     (1)     #-246     (4)
SS$_CONTROLC     00000000-XR                  119     (1)
SS$_CTRLERR      00000000-XR                  119     (1)
SS$_DATACHECK    00000000-XR                  119     (1)
SS$_NORMAL       =00000001    97      (1)   #-332     (7)
SS$_RESIGNAL     00000000-XR                  120     (1)     #-239     (4)     #-344     (7)
SYS$SETPRT       00000000-XR                  121     (1)
SYS$UNWIND       00000000-XR                  121     (1)       341     (7)
TST$MCHK         00000000-XR                  122     (1)       448     (11)      460     (11)
```

```
UBA$INITIAL       0000017E-R   409     (10)
UBA$UNEXINT       00000180-R   416     (10)
UBADP_CPU         00000171-R   383     (9)
UBINTSZ           =00000084    389     (9)
UBINT_DISP        00000000-XR                  123     (1)     #-491     (12)
UCB$L_CRB         00000020                   #-373     (8)
VEC$L_ADP         00000014                   #-374     (8)
VEC$Q_DISPATCH    00000000                   #-491     (12)      492     (12)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ADPDEF	2	83 (1)	83 (1)
$CRBDEF	1	82 (1)	82 (1)
$DEF	1	94 (1)	
$DEFINI	1	81 (1)	81 (1) 82 (1) 83 (1) 84 (1) 85 (1) 86 (1) 87 (1) 89 (1) 90 (1) 91 (1) 92 (1) 93 (1) 95 (1)
$DS_DSDEF	2	94 (1)	94 (1)
$DS_DW730_DEF	1	93 (1)	93 (1)
$DS_HPODEF	2	92 (1)	92 (1)
$DYNDEF	2	85 (1)	85 (1)
$EQU	1	94 (1)	94 (1)
$EQULS1	1	94 (1)	94 (1)
$EQULST	1		94 (1)
$GBLINI	2		94 (1)
$OFFSET	2	178 (2)	178 (2)
$OFFST1	1	182 (2)	182 (2)
$PR730DEF	1	91 (1)	91 (1)
$PRDEF	4	90 (1)	90 (1)
$PRTDEF	1	89 (1)	89 (1)
$PSLDEF	2	86 (1)	86 (1)
$PTEDEF	3	87 (1)	87 (1)
$PUSHADR	1	201 (3)	201 (3)
$UBIDEF	2	95 (1)	95 (1)
$UCBDEF	10	81 (1)	81 (1)
$VECDEF	2	84 (1)	84 (1)
$VIELD	1		88 (1)
$VIELD1	1	94 (1)	88 (1)
CASE	1	200 (3)	200 (3)
ERRSUP_S	1	201 (3)	201 (3)
MODNAM	1	132 (1)	132 (1)
SPAREVECTOR	1	538 (15)	545 (15) 546 (15) 547 (15) 548 (15) 549 (15) 550 (15) 551 (15) 552 (15) 553 (15) 554 (15) 555 (15) 556 (15) 557 (15) 558 (15) 559 (15) 560 (15) 561 (15) 562 (15) 563 (15) 564 (15) 565 (15) 566 (15) 567 (15) 568 (15) 569 (15) 570 (15) 571 (15) 572 (15) 573 (15) 574 (15) 575 (15)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.09	00:00:00.56
Command processing	136	00:00:00.70	00:00:02.13
Pass 1	1080	00:00:13.92	00:00:39.14
Symbol table sort	1	00:00:00.99	00:00:02.28
Pass 2	247	00:00:02.60	00:00:11.63
Symbol table output	28	00:00:00.18	00:00:00.43

Psect synopsis output	10	00:00:00.02	00:00:00.09
Cross-reference output	78	00:00:00.80	00:00:04.37
Assembler run totals	1615	00:00:19.31	00:01:00.63

The working set limit was 1000 pages.
76715 bytes (150 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 693 non-local and 14 local symbols.
622 source lines were read in Pass 1, producing 0 object records in Pass 2.
124 pages of virtual memory were used to define 31 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	8
DRB1:[DS.WORK]DS.MLB;218	2
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	25

978 GETS were required to define 25 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) ONLY730/UPDA=(ONLY730.UPD,ONLY730.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SY

Table of contents

(1) 100 DECLARATIONS
(2) 133 ASCII STRING PARSE ROUTINE.
(7) 486 SCAN$SYMBOL Scan symbol
(9) 583 SCAN$COMPARE Compare strings
(11) 646 SCAN$NUMERIC Scan number
(13) 790 SCAN$SEARCHLIST Search list for unique string
(15) 867 SCAN$DEVICE Scan a device name
(16) 942 Breakup file name into parts

```
                        0000     1          .TITLE  PARSE   ASCII STRING PARSE ROUTINE.
                        0000     2          .IDENT  /07-14/
                        0000     3          .NoShow Conditionals
                        0000     4
                        0000     5  ;
                        0000     6  ; Copyright (c) 1977, 1982, 1983
                        0000     7  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000     8  ;
                        0000     9  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    10  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    11  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    12  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    13  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    14  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    15  ; REMAIN IN DEC.
                        0000    16  ;
                        0000    17  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    18  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    19  ; CORPORATION.
                        0000    20  ;
                        0000    21  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    22  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    23
                        0000    24  ;++
                        0000    25  ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
                        0000    26  ;
                        0000    27  ; ABSTRACT:
                        0000    28  ;
                        0000    29  ; ENVIRONMENT:
                        0000    30  ;
                        0000    31  ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
                        0000    32  ;
                        0000    33  ; MODIFIED BY:
                        0000    34  ;                KEN CHAPMAN     02-FEB-78       VERSION 02 (ESSAA-3.07).
                        0000    35  ;       01       ADDED QUADWORD DATA CAPABILITY.
                        0000    36  ;       02       ADDED STRING MATCH FUNCTIO.
                        0000    37  ;                Roger Riggs     6-NOV-1978
                        0000    38  ;       03       ADDED KEYWORD MATCH FUNCTION AND MADE SOME SUBROUTINES
                        0000    39  ;                USEFUL FROM OTHER PARTS OF THE SUPERVISOR.
                        0000    40  ;       04       Modified keyword construct.
                        0000    41  ;                Dave Butenhof, 13-feb-1981, version 6.3
                        0000    42  ;       05       Add parse table command code for SYMBOL type, so CLI can
                        0000    43  ;                recognise $ and _ in section names.
                        0000    44  ;
                        0000    45  ;       06       - Jack Stansbury, 28-Oct-1981, Version 6.-
                        0000    46  ;                Fixed a few truncation errors. Also added .LIBRARY statements
                        0000    47  ;                for $DS and $DIAG.
                        0000    48  ;
                        0000    49  ;       07       - Dave Butenhof, 10-Nov-1981, version 6.-
                        0000    50  ;                Add new PARSE codes, CLI$K_COMMA and CLI$K_VALSEP.  The
                        0000    51  ;                former traverses the sequence "<blank>,<blank>", while the
                        0000    52  ;                latter traverses "<blank>{=!:}<blank>".  These will make
                        0000    53  ;                CLI simpler.  Added .LIBRARY for LIB, since I added CASE
                        0000    54  ;                macro useage
                        0000    55  ;
                        0000    56  ;       08       - Dave Butenhof, 11-Nov-1981, version 6.-
                        0000    57  ;                Add another PARSE code, CLI$K_EOL to check for end of
```

```
                        ;                       line (considering comments).
                        ;
                        ;       09              - Dave Butenhof, 11-Nov-1981, version 6.-
                        ;                       Again, make things easier for CLI by adding extra (internal
                        ;                       only) entry point for PARSE.  DSX$SUPERPARSE won't stop at
                        ;                       physical end of line; it'll keep parsing the null string until
                        ;                       told to stop.
                        ;
                        ;       10              - Dave Butenhof, 13-Nov-1981, version 6.5
                        ;                       One more time...add CLI$K_CALL and CLI$K_RETURN functions
                        ;                       for CLI convenience.  The parse table is just too complex.
                        ;                       Also, CLI$K_BIFS to test subparse return code
                        ;
                        ;       11              - Dave Butenhof, 03-Feb-1982, Version 6.6
                        ;                       Add Scan$File to allow wildcards (* and %) in file names.
                        ;                       This is for DIRECTORY...CLI will allow them in RUN, etc;
                        ;                       but it won't work too well.  Also, add a macro version of
                        ;                       BREAKUP_FILE (for merging default file specs) here...it's
                        ;                       much more code efficient than the bliss version.
                        ;
                        ;       12              Jack Stansbury, 15-Dec-1982, Version 6.10
                        ;                       Fixed the code in TryEol to fix a bug where a CLI$K_EOL was
                        ;                       being done twice, and causing an error when a comment
                        ;                       followed a DS command. The comment character was being
                        ;                       skipped, but not the rest of the line after the '!'. Thus,
                        ;                       the second time the TryEol was called, R5 would point to the
                        ;                       character after the '!'. Now I set R4 = 0 (the number of
                        ;                       characters remaining in the line) and set R5 to point to the
                        ;                       real end of the line (the NEWLINE character).
                        ;
                        ;       13              Bob Bergazzi    May 16,1983     Version 6.11
                        ;                       Changed the order of the .LIB statements.
                        ;
                        ;       14              M. Baggett      Nov 16,1983     Version 6.13
                        ;                       Allow device name of the form ' name$aaann' to be used as load
                        ;                       devices.
                        ;
                        ;       15              RE MUSE         MAY 8,1984      Version 7.0
                        ;                       Allow device names of the form 'node$ggan' to be used as load
                        ;                       devices
                        ;--
```

```
                     0000   100          .SBTTL  DECLARATIONS
                     0000   101 ;
                     0000   102 ; INCLUDE FILES:
                     0000   103 ;
                     0000   104          .LIBRARY        /SYS$LIBRARY:LIB/                ;                              [13]
                     0000   105          .LIBRARY        /$DS/                            ;                              [13]
                     0000   106          .LIBRARY        /$DIAG/                          ;                              [13]
                     0000   107 ;
                     0000   108 ; MACROS:
                     0000   109 ;
                     0000   110
                     0000   111 ;
                     0000   112 ; EQUATED SYMBOLS:
                     0000   113 ;
                     C000   114          $ds_clidef
                     0000   115          $DS_DSDEF
                     0000   116          $DS_PARSE_DEF
                     0000   117          DSFDEF
                     0000   118          $DS_HPODEF
00000002             0000   119 V_BIT=2
00000001             0000   120 SS$_NORMAL=1
                     0000   121
                     0000   122 ;
                     0000   123 ; OWN STORAGE:
                     0000   124 ;
                 00000000   125          .Psect  Work, Noshr, Noexe, Wrt, Long    ;                                      [10]
                     0000   126 Save_call:                                        ;                                      [10]
00000000             0000   127          .long   0                                ; Address for RETURN                   [10]
                     0004   128 Save_status:                                      ; Status of CALLed subparse            [10]
00000000             0004   129          .long   0                                ;                                      [10]
                     0008   130
                 00000000   131          .PSECT  Code, Shr, Exe, Nowrt, Byte
```

```
                        .SBTTL  ASCII STRING PARSE ROUTINE.
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS ROUTINE TAKES AN ASCII STRING INPUT AND PARSES IT USING A
                ;       CALLER SUPPLIED PARSE TREE.  UPON A MATCH IN THE TREE, IT
                ;       DISPATCHES TO A CALLER SUPPLIED ACTION ROUTINE.  UPON A MISMATCH,
                ;       IT BRANCHS TO ANOTHER POINT IN THE PARSE TREE.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       DIAGNOSTIC PROCEDURE CALL.
                ;
                ; INPUT PARAMETERS:
                ;
                ;       PARSE$_BUFADR(AP)       = INPUT COUNTED STRING ADDRESS.
                ;       PARSE$_TREE(AP)         = PARSE TREE ENTRY POINT.
                ;       PARSE$_ACTION(AP)       = SUCCESS ACTION SUBROUTINE ADDRESS.
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       RESULTS OF CALLER'S ACTION SUBROUTINE.
                ;
                ; COMPLETION CODES:
                ;
                ;       SS$_NORMAL      = STRING SUCCESSFULLY PARSED.
                ;       DS$_ERROR       = ERROR MATCH CODE ENCOUNTERED IN PARSE TREE.
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ; REGISTER USAGE:
                ;
                ;       R0      = SCRATCH AND COMPLETION CODES.
                ;       R1      = SCRATCH.
                ;       R2      = SCRATCH
                ;       R3      = RADIX AND BIT MASK POINTER
                ;       R4      = LENGTH OF REMAINING INPUT STRING
                ;       R5      = ADDRESS OF REMAINING INPUT STRING
                ;       R7      = PARSE TREE POINTER.
                ;--
```

```
                       OFFC  0000   184 .entry  dsx$superparse,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>     ;          [09]
                 56    01    D0  0002   185          movl    #1, r6                          ; Don't exit on end of line [09]
                       04    11  0005   186          brb     parse_common                    ; Join common code [09]
                       OFFC  0007   187 .ENTRY  DSX$PARSE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                       56    D4  0009   188          clrl    r6                              ; Exit on end of line [09]
                                 000B   189 parse_common:                                    ;          [09]
               54    04 BC    7D  000B   190          MOVQ    @PARSE$_BUFADR(AP),R4           ; GET LENGTH, ADDR OF INPUT STRING
                  54    54    3C  000F   191          MOVZWL  R4, R4                          ; LENGTH.
          00000000'EF    D4  0012   192          clrl    Save_call                       ; Clear saved call address [10]
               57    08 AC    D0  0018   193          MOVL    PARSE$_TREE(AP), R7             ; TREE POINTER.
                                 001C   194 TRAVERSE:
                    04 56    E8  001C   195          blbs    r6, 10$                         ; Don't check for EOL if set [09]
                       54    D5  001F   196          TSTL    R4                              ; ANY CHARACTERS LEFT IN INPUT STREAM
                       42    13  0021   197          BEQL    EXIT                            ; BRANCH IF NOT
                    F6 AF    DF  C023   198 10$:     PUSHAL  TRAVERSE                        ; FAKE A SUBROUTINE CALL. [09]
                                 0026   199          case    src=(r7), type=b, -             ; Case on the dispatch codes [07]
                                 0026   200                   limit=#cli$k_error, -           ;          [07]
                                 0026   201                   displist=< -                    ; ...dispatch list [07]
                                 0026   202                   trverr, -                       ; 0 = Error. [07]
                                 0026   203                   trvexit, -                      ; 1 = Exit. [07]
                                 0026   204                   trvbr, -                        ; 2 = Branch. [07]
                                 0026   205                   trvbif, -                       ; 3 = Conditional branch. [07]
                                 0026   206                   trvspace, -                     ; 4 = Separator characters. [07]
                                 0026   207                   trvnum, -                       ; 5 = Number string. [07]
                                 0026   208                   trvalpha, -                     ; 6 = Alpha string. [07]
                                 0026   209                   trvalnum, -                     ; 7 = Alphanumeric string. [07]
                                 0026   210                   trvoct, -                       ; 8 = Octal number string. [07]
                                 0026   211                   trvhex, -                       ; 9 = Hex number string. [07]
                                 0026   212                   trvdec, -                       ; 10 = Decimal number string. [07]
                                 0026   213                   trvstring, -                    ; 11 = Unique string. [07]
                                 0026   214                   trvkeyword, -                   ; 12 = Keyword. [07]
                                 0026   215                   trvsymbol, -                    ; 13 = Symbol. [07]
                                 0026   216                   trvcomma, -                     ; 14 = <blank>comma<blank>. [07]
                                 0026   217                   trvvalue, -                     ; 15 = <blank>{=|:}<blank>. [07]
                                 0026   218                   trvslash, -                     ; 16 = <blank>slash<blank>. [07]
                                 0026   219                   trveol, -                       ; 17 = <blank>{!|<eol>}. [08]
                                 0026   220                   trvcall, -                      ; 18 = Call subtree parse. [10]
                                 0026   221                   trvreturn, -                    ; 19 = Return from subtree. [10]
                                 0026   222                   trvbifs, -                      ; 20 = Test subparse status. [10]
                                 0026   223                   trvfile -                       ; 21 = Filename (incl. '%' and '*') [11]
                                 0026   224                   >                               ;          [07]
                                 0057   225
                                 0057   226 ;
                                 0057   227 ; NOT A SPECIAL CODE.
                                 0057   228 ;
          65    67    91  0057   229          CMPB    (R7),(R5)                       ; TREE CHAR MATCH INPUT STRING?
                30    12  005A   230          BNEQ    BRANCH                          ; BRANCH IF NOT
                55    D6  005C   231          INCL    R5                              ; UPDATE ADDRESS
                54    D7  005E   232          DECL    R4                              ; COUNT THIS CHARACTER
              0085    31  0060   233          BRW     DO_ACTION                       ; TO ACTION AND UPDATE TREE
                                 0063   234 ;
                                 0063   235 ; NORMAL EXIT CODE
                                 0063   236 ;
                                 0063   237 TRVEXIT:
                30    10  0063   238          BSBB    ACTION                          ; EXECUTE CALLER ACTION ROUTINE.
                                 0065   239 EXIT:
          50    01    D0  0065   240          MOVL    #SS$_NORMAL, R0                 ; SET NORMAL RETURN.
```

```
                    0068   241 DSX$PARSE_X:
04                  0068   242         RET                         ; RETURN.
```

```
                        0069  ;
                        0069  ; Return special code.  Note: the PARSE call mechanism is not recursive;
                        0069  ; only one level of subtree can be used.
                        0069  ;
                        0069  trvreturn:                                              ;                                  [10]
              36    10  0069          bsbb    action                                  ; Call action routine              [10]
00000004'EF   50    D0  006B          movl    r0, Save_status                         ; Save Call status for BIFS test   [10]
50  00000000'EF     D0  0072          movl    Save_call, r0                           ; Restore saved value              [10]
              09    12  0079          bneq    10$                                     ; Branch if saved address not 0    [10]
50  00660002 8F     D0  007B          movl    #ds$_error, r0                          ; If was 0, error                  [10]
              E4    11  0082          brb     dsx$parse_x                             ; Exit                             [10]
     57   50  04    C1  0084  10$:    addl3   #4, r0, r7                              ; Advance to next                  [10]
                    05  0088          rsb                                             ; Do next                          [10]
                        0089
                        0089  ;
                        0089  ; Call special code.  Note: this is not recursive.  Only one level of call
                        0089  ; may be used.  The action routine will be called before the subtree parse.
                        0089  ;
                        0089  trvcall:                                                ;                                  [10]
00000004'EF   01    D0  0089          Movl    #1, Save_status                         ; Assume successful, if not specified [10]
00000000'EF   57    D0  0090          movl    r7, Save_call                           ; Save address for return          [10]
                        0097          brb     trvbr                                   ; Fall through to TRVBR code       [10]
                        0097  ;
                        0097  ; BRANCH SPECIAL CODE.
                        0097  ;
                        0097  TRVBR:
              08    10  0097          BSBB    ACTION                                  ; EXECUTE CALLER ACTION ROUTINE.
                        0099  ;       BRB     BRANCH                                  ; FALL INTO BRANCH
                        0099  ;
                        0099  ; HERE TO TAKE BRANCH IN TREE
                        0099  ;
                        0099  BRANCH:
     50   02 A7     32  0099          CVTWL   2(R7), R0                               ; GET DISPLACEMENT.
        57   50     C0  009D          ADDL2   R0, R7                                  ; UPDATE TREE POINTER.
                    05  00A0          RSB                                             ; CONTINUE TO TRAVERSE TREE.
                        00A1  ;
                        00A1  ; DISPATCH TO CALLER'S ACTION ROUTINE.
                        00A1  ;
        5A   50     7D  00A1  ACTION: MOVQ    R0,R10                                  ; COPY QUAD NUMBER TO R10-R11
        58   55     D0  00A4          MOVL    R5,R8                                   ; COPY ADDRESS OF REMAINING STRING
        59   54     D0  00A7          MOVL    R4,R9                                   ; COPY LENGTH OF REMAINING STRING
     50   01 A7     9A  00AA          MOVZBL  1(R7), R0                               ; GET ACTION CODE.
52  00000000'EF     9E  00AE          MOVAB   L^DS$GL_CLIBASE,R2                      ; BASE CLI DATA BASE               [06]
         0080 8F    BB  00B5          PUSHR   #^M<R7>                                 ; SAVE OUR REGS
           0C BC    16  00B9          JSB     @PARSE$_ACTION(AP)                      ; GO TO ACTION ROUTINE.
         0080 8F    BA  00BC          POPR    #^M<R7>                                 ; RESTORE OUR REGS
        54   59     D0  00C0          MOVL    R9,R4                                   ; PUT CHARACTER COUNT BACK
        55   58     D0  00C3          MOVL    R8,R5                                   ; PUT CHARACTER POINT BACK
                    05  00C6          RSB
                        00C7  ;
                        00C7  ; ERROR EXIT CODE.
                        00C7  ;
                        00C7  TRVERR:
              D8    10  00C7          BSBB    ACTION                                  ; EXECUTE CALLER ACTION ROUTINE.
50  00660002 8F     D0  00C9          MOVL    #DS$_ERROR, R0                          ; SET ERROR RETURN CODE.
              96    11  00D0          BRB     DSX$PARSE_X                             ; GO EXIT.
                        00D2  ;
```

```
                        00D2   ; BIFS conditional branch code.  This is identical to BIF, except
                        00D2   ; that it tests the saved return code from a CALL subparse instead
                        00D2   ; of R0.
                        00D2   ;
                        00D2   TRVBIFS:                                        ;                                 [10]
                CD  10  00D2           Bsbb    Action                  ; Call action routine             [10]
   BE 00000004'EF  E8   00D4           Blbs    Save_status, Branch     ; Branch if OK                    [10]
                87  D5  00DB           Tstl    (r7)+                   ; Otherwise advance to next       [10]
                    05  00DD           Rsb                             ; Continue to traverse tree       [10]
                        00DE
                        00DE   ;
                        00DE   ; BIF CONDITIONAL BRANCH SPACIAL CODE.
                        00DE   ;
                        00DE   TRVBIF:
                C1  10  00DE           BSBB    ACTION                  ; EXECUTE CALLER ACTION ROUTINE.
             B6 50  E8  00E0           BLBS    R0, BRANCH              ; BRANCH IF SATISFIED.
                87  D5  00E3           TSTL    (R7)+                   ; UPDATE TREE POINTER.
                    05  00E5           RSB                             ; CONTINUE TO TRAVERSE TREE.
```

```
                                00E6    320 BRANCH_IF_NONE:
                          B1    13  00E6    321         BEQL    BRANCH                  ; Branch if Length was zero
                                00E8    322 DO_ACTION:
                          B7    10  00E8    323         BSBB    ACTION                  ; CALL ACTION ROUTINE
                          87    D5  00EA    324         TSTL    (R7)+                   ; UPDATE TREE POINTER
                                05  00EC    325         RSB                             ; CONTINUE TRAVERSE
                                    00ED    326 ;
                                    00ED    327 ; SPACE SPECIAL CODE.
                                    00ED    328 ;
                                    00ED    329 TRVSPACE:
                        012A    30  00ED    330         BSBW    SCAN$SPACE              ; CHECK FOR SPACES
                          F4    11  00F0    331         BRB     BRANCH_IF_NONE          ; TAKE BRANCH IF NO SPACES OR TABS
                                    00F2    332 ;
                                    00F2    333 ; NUMERIC VALUE CODE.
                                    00F2    334 ;
                                    00F2    335 TRVOCT:
                    53  08          D0  00F2    336         MOVL    #8, R3                  ; SET RADIX TO OCTAL.
                          11        11  00F5    337         BRB     TRVNUMA
                                00F7    338 TRVHEX:
                    53  10          D0  00F7    339         MOVL    #16, R3                 ; SET RADIX TO HEXIDECIMAL.
                          0C        11  00FA    340         BRB     TRVNUMA
                                00FC    341 TRVDEC:
                    53  0A          D0  00FC    342         MOVL    #10, R3                 ; SET RADIX TO DECIMAL.
                          07        11  00FF    343         BRB     TRVNUMA
                                0101    344 TRVNUM:
              53  00000000'EF          D0  0101    345         MOVL    L^DS$GL_RADIX, R3       ; GET DEFAULT RADIX.            [06]
                                0108    346 TRVNUMA:
                        0193    30  0108    347         BSBW    SCAN$NUMERIC            ; SCAN SOMETHING NUMERIC
                          D9    11  0108    348         BRB     BRANCH_IF_NONE          ; TAKE BRANCH IF NOTHING FOUND
                                010D    349 ;
                                010D    350 ; ALPHA CHECK.
                                010D    351 ;
                                010D    352 TRVALPHA:
                        00D4    30  010D    353         BSBW    SCAN$ALPHA              ; CHECK FOR ALPHA               [07]
                          D4    11  0110    354         BRB     BRANCH_IF_NONE          ; DO ACTION IF FOUND ELSE BRANCH
                                0112    355 ;
                                0112    356 ; ALPHA-NUMERIC CHECK.
                                0112    357 ;
                                0112    358 TRVALNUM:
                        00E1    30  0112    359         BSBW    SCAN$ALPHANUM           ; SPAN ALPHA-NUMERIC            [07]
                          CF    11  0115    360         BRB     BRANCH_IF_NONE          ; DO ACTION IF FOUND ELSE BRANCH
                                0117    361 ;
                                0117    362 ; UNIQUE STRING MATCH.
                                0117    363 ;
                                0117    364 .enabl  lsb                             ; Allow access to BRW BRANCH    [10]
                                0117    365 TRVSTRING:
              53  04 A7        9E  0117    366         MOVAB   4(R7), R3               ; GET POINTER TO ASCIC.
                52  83        9A  0118    367         MOVZBL  (R3)+, R2               ; GET ASCII COUNT.
                        0128    30  011E    368         BSBW    SCAN$COMPARE            ; COMPARE
                          37    19  0121    369         BLSS    20$                     ; TAKE BRANCH IF NO MATCH       [10]
                        FF7B    30  0123    370         BSBW    ACTION                  ; GO DO CALLER ACTION.
                          87    D5  0126    371         TSTL    (R7)+                   ; UPDATE TREE POINTER.
                50        87    9A  0128    372         MOVZBL  (R7)+, R0               ; GET ASCIC COUNT.
                57        50    C0  012B    373         ADDL2   R0, R7                  ; PAST THE ASCII.
                                05  012E    374         RSB                             ; CONTINUE TO TRAVERSE TREE.
```

```
                        012F   376 ;
                        012F   377 ; KEYWORD MATCH ROUTINE
                        012F   378 ;
                        012F   379 TRVKEYWORD:
      00A0    30        012F   380         BSBW    SCAN$SYMBOL                 ; SCAN A SYMBOL
        26    13        0132   381         BEQL    20$                         ; IF NO SYMBOL, TAKE BRANCH
   7E   50    D0        0134   382         MOVL    R0,-(SP)                    ; SAVE STRING LENGTH
53  06 A7    32        0137   383         CVTWL   6(R7),R3                    ; Get offset to keyword table
   53  57    C0        013B   384         ADDL    R7,R3                       ; Make absolute address of table
     01F9    30        013E   385         BSBW    SCAN$SEARCHLIST             ; LOCATE SYMBOL IN TABLE
        1A   14        0141   386         BGTR    30$                         ; Exit if ambigious
        0F   19        0143   387         BLSS    10$                         ; IF NOT FOUND TAKE BRANCH
04 A7   50    B1        0145   388         CMPW    R0,4(R7)                    ; CORRECT KEYWORD
        09    12        0149   389         BNEQ    10$                         ; NO, TAKE BRANCH
     FF53    30        C14B   390         BSBW    ACTION                      ; DO ACTION ROUTINE
   57   08    C0        014E   391         ADDL2   #8,R7                       ; UPDATE TREE POINTER
        8E    D5        0151   392         TSTL    (SP)+                       ; REMOVE SYMBOL LENGTH
              05        0153   393         RSB                                 ; NEXT TREE JUNCTURE
   54   6E    C0        0154   394 10$:     ADDL    (SP),R4                     ; ADJUST INPUT STRING LENGTH BY SYMBOL
   55   8E    C2        0157   395         SUBL    (SP)+,R5                    ; ADJUST INPUT STRING ADDR BY SYMBOL
     FF3C    31        015A   396 20$:     BRW     BRANCH                      ; TAKE FAILURE BRANCH
     FF05    31        015D   397 30$:     BRW     EXIT                        ; TAKE EXIT IF AMBIGIOUS
                        0160   398 .dsabl  lsb                                 ;                                          [10]
                        0160   399 ;
                        0160   400 ; Symbol match routine
                        0160   401 ;
                        0160   402 TRVSYMBOL:
        70    10        0160   403         BSBB    SCAN$SYMBOL                 ; Search for symbol
        82    11        0162   404         BRB     BRANCH_IF_NONE              ; Handle tree decision point
                        0164   405 ;
                        0164   406 ; Filename match routine
                        0164   407 ;
                        0164   408 TrvFile:                                    ;                                          [11]
      00A1    30        0164   409         BsbW    Scan$File                   ; Scan over filename                       [11]
      FF7C    31        0167   410         BrW     Branch_If_None              ; Handle tree decision point               [11]
                        016A   411 ;
                        016A   412 ; Traverse a string consisting of blanks and a comma.  If the comma is not
                        016A   413 ; found, the input string is unaffected: if the comma is found, then the
                        016A   414 ; comma and all preceding and following blanks are skipped.
                        016A   415 ;
                        016A   416 trvcomma:                                   ;                                          [07]
        53    D4        016A   417         clrl    r3                          ; Set up ',' code                          [07]
        08    11        016C   418         brb     trv_comma_value             ; Use common routine                       [07]
                        016E   419
                        016E   420 ;
                        016E   421 ; Traverse a string consisting of blanks and a slash.  If the slash is not
                        016E   422 ; found, the input string is unaffected: if the slash is found, then the
                        016E   423 ; slash and all preceding and following blanks are skipped
                        016E   424 ;
                        016E   425 trvslash:                                   ;                                          [07]
   53   01    D0        016E   426         movl    #1, r3                      ; Set up '/' code                          [07]
        03    11        0171   427         brb     trv_comma_value             ; Use common routine                       [07]
                        0173   428
                        0173   429 ;
                        0173   430 ; Traverse a string consisting of blanks and either an '=' or a ':'.  This
                        0173   431 ; is primarily for qualifier values.  The code from TRV_COMMA_VALUE on is
                        0173   432 ; common between this function and the TRVCOMMA function.
```

```
                     0173   433 ;
                     0173   434 trvvalue:
      53   02   D0   0173   435         movl    #2, r3                          ; Set up '=' code                       [07]
                     0176   436
                     0176   437 trv_comma_value:
           38   BB   0176   438         pushr   #^M<r3,r4,r5>                   ; Save current descriptor               [07]
        009F   30   0178   439         bsbw    scan$space                      ; Skip any spaces we might find         [07]
           50   D4   017B   440         clrl    r0                              ; Assume didn't find anything           [07]
           54   D5   017D   441         tstl    r4                              ; Any characters remaining?             [07]
           0E   15   017F   442         bleq    10$                             ; If not, can't be either               [07]
           54   D7   0181   443         decl    r4                              ; Anticipate finding it                 [07]
                     0183   444         case    src=(sp), type=b, -             ; Case on type code                     [07]
                     0183   445                 limit=#0, displist=< -          ;                                       [07]
                     0183   446                 10$, -                          ; 0 is ','                              [07]
                     0183   447                 20$, -                          ; 1 is '/'                              [07]
                     0183   448                 30$ -                           ; 2 is '=' or ':'                       [07]
                     0183   449                 >                               ;                                       [07]
           1D   11   018D   450         brb     60$                             ; No go                                 [07]
      2C   85   91   018F   451 10$:    cmpb    (r5)+, #^A","                   ; Compare next character to ','         [07]
           0E   11   0192   452         brb     40$                             ; Check it out                          [07]
      2F   85   91   0194   453 20$:    cmpb    (r5)+, #^A"/"                   ; Is it a '/'?                          [07]
           09   11   0197   454         brb     40$                             ; Check it out                          [07]
      3D   85   91   0199   455 30$:    cmpb    (r5)+, #^A"="                   ; Is is an '='?                         [07]
           06   13   019C   456         beql    50$                             ; Yep, OK                               [07]
   3A  FF  A5   91   019E   457         cmpb    -1(r5), #^A":"                  ; Otherwise, is it an ':'?              [07]
           08   12   01A2   458 40$:    bneq    60$                             ; If not, failure                       [07]
           74   10   01A4   459 50$:    bsbb    scan$space                      ; Skip trailing spaces, too             [07]
   04  AE   54   7D   01A6   460         movq    r4, 4(sp)                       ; Replace the saved descriptor          [07]
           50   D6   01AA   461         incl    r0                              ; We found something                    [07]
           38   BA   01AC   462 60$:    popr    #^M<r3,r4,r5>                   ; Restore registers                     [07]
           50   D5   01AE   463         tstl    r0                              ; Check if we found anything            [07]
        FF33   31   01B0   464         brw     branch_if_none                  ; And finish up command                 [07]
                     01B3   465
                     01B3   466 ;
                     01B3   467 ; TRVEOL
                     01B3   468 ; Skip spaces and tabs: if next character is an '!' or if there are no
                     01B3   469 ; more characters, then success; else failure (take MISS branch).
                     01B3   470 ;
           30   BB   01B3   471 trvEol: pushr   #^M<r4,r5>                   ; Save the input descriptor             [08]
           63   10   01B5   472         bsbb    scan$space                      ; Skip spaces if any are there          [08]
           54   D5   01B7   473         tstl    r4                              ; Check remaining length                [08]
           0C   13   01B9   474         beql    10$                             ; Branch to check                       [08]
           54   D7   01BB   475         decl    r4                              ; Downcount for "!"                     [08]
      21   85   91   01BD   476         cmpb    (r5)+, #^A"!"                   ; Otherwise, check for comment          [08]
           05   13   01C0   477         beql    10$                             ; OK, do action routine                 [08]
           30   BA   01C2   478         popr    #^M<r4,r5>                      ; Restore registers                     [08]
        FED2   31   01C4   479         brw     branch                          ; Take MISS branch                      [08]
                     01C7   480
55 55      54   C1   01C7   481 10$:    AddL3   R4,R5,R5                        ; Point R5 to the <CR> character        [12]
           54   D4   01CB   482         ClrL    R4                              ; No more characters in the line        [12]
           03   BA   01CD   483         popr    #^M<r0,r1>                      ; Remove saved registers from stack     [08]
        FF16   31   01CF   484         brw     do_action                       ; Do the action routine                 [08]
```

```
                        486          .SBTTL  SCAN$SYMBOL     Scan symbol
                        487 ;++
                        488 ; FUNCTIONAL DESCRIPTION:
                        489 ;
                        490 ;       SCAN$SYMBOL     Skips characters of the set (A-Z, a-z, 0-9, $, _) .
                        491 ;       SCAN$SPACE      Skips spaces and tabs.
                        492 ;       SCAN$ALPHA      Skips characters of the set (A-Z,a-z).
                        493 ;       SCAN$ALPHANUM   Skips characters of the set (A-z,a-z,0-9).
                        494 ;       SCAN$SPANC      Skips characters of the set specified by R3
                        495 ;
                        496 ; CALLING SEQUENCE:
                        497 ;
                        498 ;       BSBW    SCAN$SYMBOL
                        499 ;       BSBW    SCAN$ALPHA
                        500 ;       BSBW    SCAN$ALPHANUM
                        501 ;       BSBW    SCAN$SPACE
                        502 ;       BSBW    SCAN$SPANC
                        503 ;
                        504 ; INPUT PARAMETERS:
                        505 ;
                        506 ;       R3      Address of bit mask for valid characters (SCAN$SPANC only)
                        507 ;       R4      Length of input string
                        508 ;       R5      Address of input string
                        509 ;
                        510 ; IMPLICIT INPUTS:      NONE
                        511 ;
                        512 ; OUTPUT PARAMETERS:
                        513 ;
                        514 ;       R0      Length of symbol
                        515 ;       R1      Address of symbol
                        516 ;       R3      Address of the bit mask used for validity
                        517 ;       R4      Updated length of input string
                        518 ;       R5      Updated address of input string
                        519 ;
                        520 ; IMPLICIT OUTPUTS:      NONE
                        521 ;
                        522 ; COMPLETION CODES:      N/A
                        523 ;
                        524 ; SIDE EFFECTS:         NONE
                        525 ;
                        526 ; REGISTER USAGE:
                        527 ;
                        528 ;       R0      Length of symbol
                        529 ;       R1      Address of symbol
                        530 ;       R3      Address of bit mask used for validity
                        531 ;       R4      Current remaining length of input string
                        532 ;       R5      Current address of input string
                        533 ;
                        534 ; CONDITION CODES:
                        535 ;
                        536 ;       Z-bit   Set if length of symbol=0, else set
                        537 ;       N-bit   Zero
                        538 ;--
```

```
                                        01D2    540          .ENABL  LSB
                                        01D2    541 SCAN$SYMBOL::
                                 58  10 01D2    542          BSBB    10$                     ; Address Mask and branch to common
                                        01D4    543 DS$GT_VSYMBOL::
07FFFFFE 87FFFFFE 03FF0010 00000000     01D4    544          .LONG   ^X00000000,^X03FF0010,^X87FFFFFE,^X07FFFFFE
                                        01E4    545
                                        01E4    546 SCAN$ALPHA::
                                 46  10 01E4    547          BSBB    10$                     ; Address Mask and branch to common
                                        01E6    548 DS$GT_VALPHA::
07FFFFFE 07FFFFFE 00000000 00000000     01E6    549          .LONG   ^X00000000,^X00000000,^X07FFFFFE,^X07FFFFFE
                                        01F6    550
                                        01F6    551 SCAN$ALPHANUM::
                                 34  10 01F6    552          BSBB    10$                     ; Address Mask and branch to common
                                        01F8    553 DS$GT_VALPHANUM::
07FFFFFE 07FFFFFF 03FF0000 00000000     C1F8    554          .LONG   ^X00000000,^X03FF0000,^X07FFFFFE,^X07FFFFFE
                                        0208    555
                                        0208    556 Scan$File::
                                 22  10 0208    557          BsbB    10$                     ; Address Mask and branch to common
                                        020A    558 Ds$GT_VFile::
07FFFFFE 07FFFFFE 03FF0420 00000000     020A    559          .Long   ^X00000000,^X03FF0420,^X07FFFFFE,^X07FFFFFE
                                        021A    560
                                        021A    561 SCAN$SPACES::
                                        021A    562 SCAN$SPACE::
                                 10  10 021A    563          BSBB    10$                     ; Address Mask and branch to common
                                        021C    564 DS$GT_VSPACE::
00000000 00000000 00000001 00000200     021C    565          .LONG   ^X00000200,^X00000001,^X00000000,^X00000000
                                        022C    566
                                 08  BA 022C    567 10$:     POPR    #^M<R3>                 ; Get address of mask to R3
                                        022E    568                                         ; Fall into SCAN$SPANC
                                        022E    569          .DSABL  LSB
                                        022E    570
                                        022E    571 SCAN$SPANC::
                              54    54  3C 022E    572          MOVZWL  R4,R4                   ; Trim to just length
                              51    55  D0 0231    573          MOVL    R5,R1                   ; Copy start of string
                                    09  11 0234    574          BRB     20$                     ; Start with count of byte
                              50    65  9A 0236    575 10$:     MOVZBL  (R5),R0                 ; Fetch a character
                           05 63        E1 0239    576          BBC     R0,(R3),30$             ; Branch if not valid
                                    55  D6 023D    577          INCL    R5                      ; Update address
                                 F4 54  F4 023F    578 20$:     SOBGEQ  R4,10$                  ; Decrement count and branch if more
                                    54  D6 0242    579 30$:     INCL    R4                      ; Fix count of characters remaining
                           50   55  51  C3 0244    580          SUBL3   R1,R5,R0                ; Calculate the length just scanned
                                        05 0248    581 40$:     RSB                             ; Return
```

```
0249    583          .SBTTL  SCAN$COMPARE    Compare strings
0249    584 ;++
0249    585 ; FUNCTIONAL DESCRIPTION:
0249    586 ;
0249    587 ;       This routine performs the same function as the CMPC3
0249    588 ;       machine instruction but does not use the instructions.
0249    589 ;
0249    590 ; CALLING SEQUENCE:
0249    591 ;
0249    592 ;       BSBW    SCAN$COMPARE
0249    593 ;
0249    594 ; INPUT PARAMETERS:
0249    595 ;
0249    596 ;       R2      Length of string 1
0249    597 ;       R3      Address of string1
0249    598 ;       R4      Length of string 2
0249    599 ;       R5      Address of string 2
0249    600 ;
0249    601 ; IMPLICIT INPUTS:      NONE
0249    602 ;
0249    603 ; OUTPUT PARAMETERS:
0249    604 ;
0249    605 ;       R0      Length of matching section
0249    606 ;       R1      Address of start of string1
0249    607 ;       R2      Length of uncompared string 1
0249    608 ;       R3      Address of Uncompared byte in string 1
0249    609 ;       R4      Length of Uncompared string 2
0249    610 ;       R5      Address of uncompared byte in string 2
0249    611 ;
0249    612 ; IMPLICIT OUTPUTS:     NONE
0249    613 ;
0249    614 ; SIDE EFFECTS:         NONE
0249    615 ;
0249    616 ; COMPLETION CODES:     N/A
0249    617 ;
0249    618 ; CONDITION CODES:
0249    619 ;
0249    620 ;       Z-bit   Set if Exact match, either string1 or string exhausted
0249    621 ;       N-bit   Set if NO characters matched
0249    622 ;       C-bit   unpredictable
0249    623 ;       V-bit   unpredictable
0249    624 ;--
```

```
                        0249    626 SCAN$COMPARE::
54      54      3C      0249    627             MOVZWL  R4,R4                  ; Trim length
        50      D4      024C    628             CLRL    R0                    ; Clear length of matched string
        52      D5      024E    629             TSTL    R2                    ; Any in string1?
        15      13      0250    630             BEQL    20$                   ; Branch if none
        54      D5      0252    631 10$:        TSTL    R4                    ; Any in string2?
        11      13      0254    632             BEQL    20$                   ; Branch if string2 exhausted
65      63      91      0256    633             CMPB    (R3),(R5)             ; Compare a byte
        0C      12      0259    634             BNEQ    20$                   ; Branch if no match
        50      D6      025B    635             INCL    R0                    ; Count character in match
        53      D6      025D    636             INCL    R3                    ; Update string1 address
        55      D6      025F    637             INCL    R5                    ; Update string2 address
                        0261    638
        54      D7      0261    639             DECL    R4                    ; Count down characters in string2
        52      D7      C263    640             DECL    R2                    ; Count down characters in string1
        EB      14      0265    641             BGTR    10$                   ; Branch if string1 NOT exhausted
                        0267    642 20$:
01      50      D1      0267    643             CMPL    R0,#1                 ; Set condition codes from match length
                05      026A    644             RSB
```

```
                        026B   646          .SBTTL  SCAN$NUMERIC    Scan number
                        026B   647 ;++
                        026B   648 ; FUNCTIONAL DESCRIPTION:
                        026B   649 ;
                        026B   650 ;       This routine can be called to scan and convert a ascii
                        026B   651 ;       number to a signed, quadword value:
                        026B   652 ;       The input form is: [+!-] [% B!b!D!d!O!o!X!x] [+!-]<digit-list>
                        026B   653 ;       If the radix override is not present then the input radix is used for
                        026B   654 ;       conversion.
                        026B   655 ;
                        026B   656 ; CALLING SEQUENCE:
                        026B   657 ;
                        026B   658 ;       BSBW    SCAN$NUMERIC
                        026B   659 ;
                        026B   660 ; INPUT PARAMETERS:
                        026B   661 ;
                        026B   662 ;       R3      Radix for conversion
                        026B   663 ;       R4      Length of input string
                        026B   664 ;       R5      Address of input string
                        026B   665 ;
                        026B   666 ; IMPLICIT INPUTS:      NONE
                        026B   667 ;
                        026B   668 ; OUTPUT PARAMETERS:
                        026B   669 ;
                        026B   670 ;       R0/R1   Signed Quadword value, both zero if no conversion
                        026B   671 ;       R3      Updated conversion radix
                        026B   672 ;       R4      Updated length of string
                        026B   673 ;       R5      Updated address of string
                        026B   674 ;
                        026B   675 ; CONDITION CODES:
                        026B   676 ;
                        026B   677 ;       Z-bit   Set if No digits scanned
                        026B   678 ;       N-Bit   Unpredictable
                        026B   679 ;       C-bit   Unpredictable
                        026B   680 ;       V-bit   Unpredictable
                        026B   681 ;
                        026B   682 ; IMPLICIT OUTPUTS:     NONE
                        026B   683 ;
                        026B   684 ; COMPLETION CODES:     N/A
                        026B   685 ;
                        026B   686 ; SIDE EFFECTS:         NONE
                        026B   687 ;
                        026B   688 ; REGISTER USAGE:
                        026B   689 ;
                        026B   690 ;       R0/R1   Signed quadword value
                        026B   691 ;       R2      Current character
                        026B   692 ;       R3      Radix for conversion
                        026B   693 ;       R4      Length remaining ir string
                        026B   694 ;       R5      Address of next avail character
                        026B   695 ;--
                        026B   696
                        026B   697 DS$GT_VHEX::
0000007E 0000007E 03FF0000 00000000    026B   698          .LONG   0,^X03FF0000,^X7E,^X7E
                        027B   699 BASE_LIST:
08 0A 02 00 78 6F 64 62 58 4F 44 42    027B   700          .ASCII  "BDOXbdox"<0><2><10><8><16><2><10><8><16>
                10 08 0A 02 10         0287
```

```
                     028C   702 SCAN$DECIMAL::
53   0A   D0  028C   703         MOVL    #10,R3                  ; Set radix
     0D   11  028F   704         BRB     SCAN$NUMERIC
              0291   705
              0291   706 SCAN$OCTAL::
53   08   D0  0291   707         MOVL    #8,R3                   ; Set radix
     08   11  0294   708         BRB     SCAN$NUMERIC
              0296   709
              0296   710 SCAN$BINARY::
53   02   D0  0296   711         MOVL    #2,R3                   ; Set radix
     03   11  0299   712         BRB     SCAN$NUMERIC
              029B   713
              029B   714 SCAN$HEX::
53   10   D0  029B   715         MOVL    #16,R3                  ; Set radix
              C29E   716
              029E   717 SCAN$NUMERIC::
54   54   3C  029E   718         MOVZWL  R4,R4                   ; Trim length
     30   BB  02A1   719         PUSHR   #^M<R4,R5>              ; Save in case have to backup
              02A3   720
     5D   10  02A3   721         BSBB    CHECK_SIGN              ; Check for sign value
     50   DD  02A5   722         PUSHL   R0                      ; Save sign, 0=positive,1=negative
     70   10  02A7   723         BSBB    CHECK_RADIX             ; Set R3 from possible radix change
     57   10  02A9   724         BSBB    CHECK_SIGN              ; Check for sign value again
6E   50   C8  02AB   725         BISL    R0,(SP)                 ; Set negative if sign encountered
              02AE   726
              02AE   727 CHECK_DIGIT:
     55   DD  02AE   728         PUSHL   R5                      ; Save current position
     50   7C  02B0   729         CLRQ    R0                      ; Clear accumulator
     2D   11  02B2   730         BRB     70$                     ; Start by counting characters
52   65   9A  02B4   731 40$:    MOVZBL  (R5),R2                 ; Get the character
28 B0 AF  52  E1  02B7   732         BBC     R2,DS$GT_VHEX,80$   ; Finish if not valid digit
   03 52  06  E1  02BC   733         BBC     #6,R2,50$           ; Branch if not A-F
52   09   C0  02C0   734         ADDL    #9,R2                   ; Offset "A" to %XA
52   F0 8F  8A  02C3   735 50$:    BICB    #^C15,R2                ; Clear extra bits
53   52   D1  02C7   736         CMPL    R2,R3                   ; Compare with radix
     18   18  02CA   737         BGEQ    80$                     ; Assume end of number
     55   D6  02CC   738         INCL    R5                      ; Update string address
7E   51   53  C5  02CE   739         MULL3   R3,R1,-(SP)             ; Radix * high order
     50   D5  02D2   740         TSTL    R0                      ; If low order sign bit set
     03   18  02D4   741         BGEQ    60$                     ; Branch if not
6E   53   C0  02D6   742         ADDL    R3,(SP)                 ; Compinsate for unsiged arith
50   52   50  53  7A  02D9   743 60$:    EMUL    R3,R0,R2,R0             ; Radix * old number + new digit
     51   8E  C0  02DE   744         ADDL    (SP)+,R1                ; Include extra high part
        D0 54  F4  02E1   745 70$:    SOBGEQ  R4,40$                  ; Branch if another character present
     54   D6  02E4   746 80$:    INCL    R4                      ; Restore count
     04   BA  02E6   747         POPR    #^MR2                   ; Get position when started digits
        08 8E  E9  02E8   748         BLBC    (SP)+,90$               ; Branch if No sign or "+" sign
50   50   D2  02EB   749         MCOML   R0,R0                   ; Complement Low order
51   51   D2  02EE   750         MCOML   R1,R1                   ; Complement High order
     50   D6  02F1   751         INCL    R0                      ; Inc Low order = Negated
51   00   D8  02F3   752         ADWC    #0,R1                   ; Inc High order with Carry from low
5E   08   C0  02F6   753 90$:    ADDL    #8,SP                   ; Remove R4/R5 from stack
55   52   D1  02F9   754         CMPL    R2,R5                   ; Set Z-bit from number of digits
     05   02FC   755         RSB                             ; Return to caller
              02FD   756 INVALID:
     50   7C  02FD   757         CLRQ    R0                      ; Clear R0/R1, clear V-bit
     30   BA  02FF   758         POPR    #^M<R4,R5>              ; Restore registers
```

```
                     05  0301    759          RSB                             ; Return to caller
                         0302    760
                         0302    761 CHECK_SIGN:
                 50  D4  0302    762          CLRL    R0                      ; Assume positive
                 54  D5  0304    763          TSTL    R4                      ; Check for sign character
                 10  15  0306    764          BLEQ    40$                     ; Branch if no sign, therefore no digits
              65 2B  91  0308    765          CMPB    #^A"+",(R5)             ; Positive?
                 07  13  030B    766          BEQL    30$                     ; Yes, use and skip
              65 2D  91  030D    767          CMPB    #^A"-",(R5)             ; Negative?
                 06  12  0310    768          BNEQ    40$                     ; No, Not sign
                 50  D7  0312    769          DECL    R0                      ; Mark negative
                         0314    770 30$:
                 55  D6  0314    771          INCL    R5                      ; Remove character
                 54  D7  0316    772          DECL    R4                      ; Remove character
                         0318    773 40$:
                     05  0318    774          RSB                             ; Return
                         0319    775
                         0319    776 CHECK_RADIX:
              02 54  D1  0319    777          CMPL    R4,#2                   ; Enough chars for radix mod?
                 1B  19  031C    778          BLSS    30$                     ; No, skip check
              65 25  91  031E    779          CMPB    #^A"%",(R5)             ; Start of radix modifier?
                 16  12  0321    780          BNEQ    30$                     ; No, Finish and exit
           52 FF54 CF  9E  0323    781          MOVAB   BASE_LIST,R2            ; Address of valid base list
            01 A5  82  91  0328    782 10$:     CMPB    (R2)+,1(R5)             ; match?
                 FA  1F  032C    783          BLSSU   10$                     ; Continue unless end of table
                 09  12  032E    784          BNEQ    30$                     ; Return invalid if no match here
           53 08 A2  9A  0330    785 20$:     MOVZBL  8(R2),R3                ; Fetch modified base
              54 02  C2  0334    786          SUBL2   #2,R4                   ; Decrement by characters used
                 85  B5  0337    787          TSTW    (R5)+                   ; Pass over characters used
                     05  0339    788 30$:     RSB                             ; Return
```

```
                        790          .SBTTL  SCAN$SEARCHLIST Search list for unique string
                        791 ;++
                        792 ; FUNCTIONAL DESCRIPTION:
                        793 ;
                        794 ;       This routine can be used to validate a keyword in a list of the form:
                        795 ;       .WORD   count,offset-item1,offset-item2.....
                        796 ;       The offsets are self relative, i.e. ITEM-.
                        797 ;
                        798 ; CALLING SEQUENCE:
                        799 ;
                        800 ;       BSBW    SCAN$SEARCHLIST
                        801 ;
                        802 ; INPUT PARAMETERS:
                        803 ;
                        804 ;       R0      Length of string to be searched for
                        805 ;       R1      Address of string to be search for
                        806 ;       R3      Address of Keyword table
                        807 ;
                        808 ; IMPLICIT INPUTS:      NONE
                        809 ;
                        810 ; OUTPUT PARAMETERS:
                        811 ;
                        812 ;       R0      Index of first string that matched
                        813 ;       R1      Address of ASCIC string that matched
                        814 ;       R2-R5   Preserved across call
                        815 ;
                        816 ; IMPLICIT OUTPUTS:      NONE
                        817 ;
                        818 ; COMPLETION CODES:      N/A
                        819 ;
                        820 ; CONDITION CODES:
                        821 ;
                        822 ;       Z-Bit   Set if exactly 1 match
                        823 ;       N-bit   Set if no matchs
                        824 ;       V-bit   zero
                        825 ;       C-bit   zero
                        826 ;
                        827 ; SIDE EFFECTS:          NONE
                        828 ;--
```

B 1 DSX$EndSub Routine
C 1 DSX$EndSub Routine
D 1 DSX$EndSub Routine
E 1 DSX$CkLoop Routine
F 1 DSX$CkLoop Routine
G 1 DSX$InLoop Routine
H 1 DSX$InLoop Routine
I 1 Symbol table
J 1 Symbol table
K 1 Symbol table
L 1 Psect synopsis
M 1 Cross reference
N 1 Cross reference
B 2 Cross reference
C 2 Cross reference
D 2 Cross reference
E 2 Cross reference
F 2 Cross reference
G 2 Cross reference
H 2 - MASSBUS ADAPTER INTERRUPT DI
I 2 - MASSBUS ADAPTER INTERRUPT DI
J 2 - MASSBUS ADAPTER INTERRUPT DI
K 2 MASSBUS ADAPTER INTERRUPT DISP
L 2 MASSBUS ADAPTER INTERRUPT DISP
M 2 MASSBUS ADAPTER INITIALIZATION
N 2 Symbol table
B 3 Symbol table
C 3 Cross reference
D 3 Cross reference
E 3 *** MCHK730 machine check form
F 3 *** MCHK730 machine check form
G 3 CHK$MCHK
H 3 CHK$MCHK
I 3 CHK$MCHK
J 3 CHK$MCHK
K 3 CHK$MCHK
L 3 DSR$FAULT_CLEAR
M 3 DSR$FAULT_CLEAR
N 3 DSR$FAULT_CLEAR
B 4 *** MEMALC dynamic memory allo
C 4 *** MEMALC dynamic memory allo
D 4 *** MEMALC dynamic memory allo
E 4 *** MEMALC dynamic memory allo
F 4 *** MEMALC dynamic memory allo
G 4 ALLOCATE MEMORY AND CONDITIONA
H 4 ALLOCATE MEMORY AND CONDITIONA
I 4 ALLOCATE MEMORY AND CONDITIONA
J 4 ALLOCATE NONPAGED DYNAMIC MEMO
K 4 ALLOCATE NONPAGED DYNAMIC MEMO
L 4 GENERAL ALLOCATE MEMORY SUBRO
M 4 DEALLOCATE NONPAGED DYNAMIC ME
N 4 CHECK BLOCK PARAMETERS SUBROUT
B 5 GENERAL DEALLOCATION SUBROUTIN
C 5 MEMPOOL$INIT ;USER MODE MEMO
D 5 MEMPOOL$INIT ;USER MODE MEMO
E 5 Symbol table
F 5 Symbol table
G 5 Cross reference
H 5 Cross reference
I 5 Cross reference
J 5 Cross reference
K 5 *** MEMMGT Memory management s
L 5 *** MEMMGT Memory management s
M 5 *** MEMMGT Memory management s
N 5 *** MEMMGT Memory management s
B 6 Libraries, Macros, and Externa
C 6 Own Storage
D 6 Data Storage
E 6 MapMem Routine - Map all of me
F 6 MapMem Routine - Map all of me
G 6 MapMem Routine - Map all of me
H 6 MapMem Routine - Map all of me
I 6 MapMem Routine - Map all of me
J 6 MapMem Routine - Map all of me
K 6 MAPMEM$IOSPACE Setup page tabl
L 6 MAPMEM$IOSPACE Setup page tabl
M 6 ACCESSABLE check a page for ac
N 6 STACKPROT
B 7 MAP FREE MEMORY PROCEDURE.
C 7 MAP FREE MEMORY PROCEDURE.
D 7 GET A BLOCK OF VIRTUAL MEMORY.
E 7 GET A BLOCK OF VIRTUAL MEMORY.
F 7 GET A BLOCK OF VIRTUAL MEMORY.
G 7 RELEASE A BLOCK OF VIRTUAL MEM
H 7 RELEASE A BLOCK OF VIRTUAL MEM
I 7 RELEASE A BLOCK OF VIRTUAL MEM
J 7 TURN MEMORY MANAGEMENT ON.
K 7 TURN MEMORY MANAGEMENT OFF.
L 7 DSR$MMENABLE Enable to disable
M 7 SET PROTECTION ON PAGES.
N 7 SET PROTECTION ON PAGES.
B 8 SET PROTECTION ON PAGES.
C 8 SET PROTECTION ON PAGES.
D 8 CALCULATE PTE ADDRESS SUBROUTI
E 8 CALCULATE PTE ADDRESS SUBROUTI
F 8 DSX$MAPDBGBLOCK - MAP A BLOCK
G 8 DSX$FREEDBGSYM - FREE MEMORY M
H 8 MAP_DEBUGGER - MAP THE DEBUGGE
I 8 UNMAP_DEBUGGER - UNMAP THE DEB
J 8 UNMAP_DEBUGGER - UNMAP THE DEB
K 8 UNMAP_DEBUGGER - UNMAP THE DEB
L 8 Symbol table
M 8 Symbol table
N 8 Symbol table
B 9 Symbol table
C 9 Symbol table
D 9 Cross reference
E 9 Cross reference
F 9 Cross reference
G 9 Cross reference
H 9 Cross reference
I 9 Cross reference
J 9 Cross reference
K 9 Cross reference
L 9 Cross reference
M 9 - TU81 LOAD DRIVER
N 9 - TU81 LOAD DRIVER
B 10 - TU81 LOAD DRIVER
C 10 DECLARATIONS
D 10 DECLARATIONS
E 10 DECLARATIONS
F 10 TU81 load device initializatio
G 10 TU81 load device initializatio
H 10 TU81 load device initializatio
I 10 TU81 load driver QIO
J 10 TU81 load driver QIO
K 10 TU81 load driver QIO
L 10 TU81 load driver QIO
M 10 Symbol table
N 10 Psect synopsis
B 11 Cross reference
C 11 Cross reference
D 11 Cross reference
E 11 Cross reference
F 11 *** ODS Disk RMS routines
G 11 declarations
H 11 declarations
I 11 Advance RFA
J 11 Advance RFA
K 11 Access file block
L 11 Access file block
M 11 Access file block
N 11 Access file block
B 12 ODS OPEN service
C 12 ODS OPEN service
D 12 ODS OPEN service
E 12 ODS$GET routine
F 12 ODS$GET routine
G 12 ODS$GET routine
H 12 ODS$GET routine
I 12 ODS$GET routine
J 12 ODS$READ routine
K 12 ODS$READ routine
L 12 ODS$READ routine
M 12 ODS$READ routine
N 12 *** ONLY730 NEBULA specific ro
B 13 *** ONLY730 NEBULA specific ro
C 13 *** ONLY730 NEBULA specific ro
D 13 Macro invocations
E 13 Macro invocations
F 13 DSP$CONFIG_ADP Build P-table f
G 13 DSP$CONFIG_ADP Build P-table f
H 13 DSP$CONFIG_ADP Build P-table f
I 13 MAP$IOSPACE Validate mapping f
J 13 MAP$IOSPACE Validate mapping f
K 13 FETCH_LONG Retreive longword f
L 13 PURGE_DATAPATH
M 13 PURGE_DATAPATH
N 13 UBA$INITIAL UNIBUS ADAPTER INI
B 14 IO$TESTCSR_x Test CSR for exis
C 14 IOGEN$CONN_VEC
D 14 QIOCLEAN_CPU
E 14 ONLY_SCB_CPU specific SCB setu
F 14 SCBVECTOR_xxx Otherwise undefi
G 14 SCBVECTOR_xxx Otherwise undefi
H 14 ONLY_CLOCK_INIT Perform cpu-sp
I 14 ONLY_CLOCK_INIT Perform cpu-sp
J 14 Symbol table
K 14 Symbol table
L 14 Psect synopsis

```
M 14  Cross reference
N 14  Cross reference
B 15  Cross reference
C 15  Cross reference
D 15  Cross reference
E 15  Cross reference
F 15  ASCII STRING PARSE ROUTINE.
G 15  ASCII STRING PARSE ROUTINE.
H 15  ASCII STRING PARSE ROUTINE.
I 15  DECLARATIONS
J 15  ASCII STRING PARSE ROUTINE.
K 15  ASCII STRING PARSE ROUTINE.
L 15  ASCII STRING PARSE ROUTINE.
M 15  ASCII STRING PARSE ROUTINE.
N 15  ASCII STRING PARSE ROUTINE.
B 16  ASCII STRING PARSE ROUTINE.
C 16  ASCII STRING PARSE ROUTINE.
D 16  ASCII STRING PARSE ROUTINE.
E 16  SCAN$SYMBOL Scan symbol
F 16  SCAN$SYMBOL Scan symbol
G 16  SCAN$COMPARE Compare strings
H 16  SCAN$COMPARE Compare strings
I 16  SCAN$NUMERIC Scan number
J 16  SCAN$NUMERIC Scan number
K 16  SCAN$NUMERIC Scan number
L 16  SCAN$SEARCHLIST Search list fo
```

```
                              033A  SCAN$SEARCHLIST::
            50     50   3C   033A            MOVZWL  R0,R0                   ; Trim length
                   3F   BB   033D            PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
                   7E   7C   033F            CLRQ    -(SP)                   ; Space for Index 0(SP), Address 4(SP)
                   7E   D4   0341            CLRL    -(SP)                   ; Count of matches
                             0343
            52     83   3C   0343            MOVZWL  (R3)+,R2                ; Get count of strings to match
         53      6342   3E   0346            MOVAW   (R3)[R2],R3             ; Point past last word
                   2E   11   034A            BRB     60$                     ; Start by counting
                             034C  10$:
            55     73   32   034C            CVTWL   -(R3),R5                ; Get offset to ASCIC string
                   29   13   034F            BEQL    60$                     ; Skip it if offset=0
            55     53   C0   0351            ADDL    R3,R5                   ; Make it absolute
            54     85   9A   0354            MOVZBL  (R5)+,R4                ; Fetch length of ASCIC string
                   21   13   0357            BEQL    60$                     ; Branch if null string, try next
                             0359  20$:
        50      0C AE   7D   0359            MOVQ    12(SP),R0               ; Restore original len,addr
            54     50   D1   035D            CMPL    R0,R4                   ; compare
                   18   14   0360            BGTR    60$                     ; Branch if input longer than string
                   05   11   0362            BRB     40$                     ; Start by counting character
                             0364  30$:
            85     81   91   0364            CMPB    (R1)+,(R5)+             ; Compare a byte
                   11   12   0367            BNEQ    60$                     ; Mismatch, try next string
                F8 50   F4   0369  40$:      SOBGEQ  R0,30$                  ; Branch if more characters in input
                   6E   D6   036C            INCL    (SP)                    ; Count this match
        04 AE      52   D0   036E            MOVL    R2,4(SP)                ; Save index of string match
            55     63   32   0372            CVTWL   (R3),R5                 ; Get offset to ASCIC
  08 AE     55     53   C1   0375            ADDL3   R3,R5,8(SP)             ; Save address of match entry
                             037A  60$:
                CF 52   F4   037A            SOBGEQ  R2,10$                  ; If not last string, do next
                             037D
            50     8E   D0   037D            MOVL    (SP)+,R0                ; Get count of matches
            6E     8E   7D   0380            MOVQ    (SP)+,(SP)              ; Remove saved R0/R1
            01     50   D1   0383            CMPL    R0,#1                   ; set condition codes from # of matchs
                   3F   BA   0386            POPR    #^M<R0,R1,R2,R3,R4,R5>  ; index, address, restore R2,R3,R4,R5
                        05   0388            RSB                             ; Return
```

```
                                .SBTTL  SCAN$DEVICE     Scan a device name
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine can be used to scan a device name of the form:
                ;       ggan[:] , gg is generic device name a is single letter A-z,
                ;       and n is a decimal number in the range 0-255.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    SCAN$DEVICE
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R4      Length of input string to scan
                ;
                ;       R5      Address of input string to be scanned
                ;
                ; IMPLICIT INPUTS:      NONE
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R0      <0,16> length of name scanned, <16,16> unit number
                ;       R1      Address of device name
                ;       R4      Updated input string length
                ;       R5      Updated input string address
                ;
                ; COMPLETION CODES:
                ;
                ; CONDITION CODES:
                ;
                ;       Z-bit   Set if no device present
                ;
                ; IMPLICIT OUTPUTS:     NONE
                ;
                ; SIDE EFFECTS:         NONE
                ;
                ;--
```

```
                                0389    906 SCAN$DEVICE::
                          54    3C  0389    907         MOVZWL  R4,R4                   ; Trim length
                          30    BB  038C    908         PUSHR   #^M<R4,R5>             ; Save input pointers
                          30    BB  038E    909         PUSHR   #^M<R4,R5>             ; Save input pointers
                       FE63     30  0390    910         BSBW    SCAN$ALPHANUM          ; check name                        [15]
                       24 65    91  0393    911         CMPB    (R5),#^A'$'            ; is this a long device name ?      [15]
                          08    12  0396    912         BNEQ    5$                     ; if not then continue              [15]
                          55    D6  0398    913         INCL    R5                     ; else skip '$'                     [15]
                          54    D7  039A    914         DECL    R4                     ; skip character                    [15]
                          8E    7C  039C    915         CLRQ    (SP)+                  ; reset stack                       [15]
                          02    11  039E    916         BRB     7$                     ; go check generic                  [15]
                                03A0    917 5$:                                        ;                                   [15]
                          30    BA  03A0    918         POPR    #^M<R4,R5>             ; restore original registers        [15]
                                03A2    919 7$:                                        ;                                   [15]
                       FE3F     30  03A2    920         BSBW    SCAN$ALPHA             ; Get an alpha field
                          23    13  03A5    921         BEQL    40$                    ; Branch if not present
                       FEE2     30  03A7    922         BSBW    SCAN$DECIMAL           ; Scan a decimal number
                          03    12  03AA    923         BNEQ    10$                    ; Branch if number scanned
                       50 01    CE  03AC    924         MNEGL   #1,R0                  ; Force sign bit on
                 6E    55 04 AE    C3  03AF    925 10$:    SUBL3   4(SP),R5,(SP)          ; Length of name is new position- old
                    02 AE    50    B0  03B4    926         MOVW    R0,2(SP)               ; Save unit number in top of length
                          54    D7  03B8    927         DECL    R4                     ; Subtract for length of ":"
                          07    19  03BA    928         BLSS    20$                    ; Branch if no character there
                       3A 85    91  03BC    929         CMPB    (R5)+,#^A":"           ; Check for optional ":"
                          04    13  03BF    930         BEQL    30$                    ; Branch if present
                          55    D7  03C1    931         DECL    R5                     ; Fix address
                          54    D6  03C3    932 20$:    INCL    R4                     ; Fix length
                          03    BA  03C5    933 30$:    POPR    #^M<R0,R1>             ; Get length, address of device name
                          50    B5  03C7    934         TSTW    R0                     ; Set condition codes, from length
                                05  03C9    935         RSB                            ; Return to caller
                                    03CA    936 40$:
                          50    7C  03CA    937         CLRQ    R0                     ; Clear length, address
                          30    BA  03CC    938         POPR    #^M<R4,R5>             ; restore pointers
                                05  03CE    939         RSB                            ; Return
                                    03CF    940
```

```
                                03CF    942 .Subtitle       Breakup File name into parts
                                03CF    943
                                03CF    944 ;++
                                03CF    945 ; Function:
                                03CF    946 ;       Break up fields of a filename spec so that they can be merged by
                                03CF    947 ;       repeated calls to this routine.  If any of the filespec fields are
                                03CF    948 ;       present, the descriptor for that field is overlayed over the
                                03CF    949 ;       appropriate field of the output file block
                                03CF    950 ;
                                03CF    951 ; Inputs:
                                03CF    952 ;       04(Ap)  Length of filename string
                                03CF    953 ;       08(Ap)  Address of filename string
                                03CF    954 ;       12(Ap)  Address of file block:
                                03CF    955 ;                       (00) :  Descriptor of device part (incl. ':')
                                03CF    956 ;                       (08) :  Descriptor of directory part (incl. '[' & ']')
                                03CF    957 ;                       (16) :  Descriptor of filename part
                                03CF    958 ;                       (24) :  Descriptor of file type part (incl. '.')
                                03CF    959 ;                       (32) :  Descriptor of version part (incl. ';')
                                03CF    960 ;
                                03CF    961 ;
                                03CF    962 ;--
                                03CF    963
                                03CF    964 OverLay:                                    ; Utility routine                   [11]
52  54  50  C3                 03CF    965         SubL3   R0, R4, R2                  ; Calculate length of field         [11]
        52  D6                 03D3    966         IncL    R2                          ; .. include terminator             [11]
    53  55  D0                 03D5    967         MovL    R5, R3                      ; Field starts at 'pivot'           [11]
  6647  52  7D                 03D8    968         MovQ    R2, (R6)[R7]                ; Move to correct descriptor        [11]
55  51  C1  C1                 03DC    969         AddL3   #1, R1, R5                  ; Update string pointer             [11]
54  50  01  C3                 03E0    970         SubL3   #1, R0, R4                  ; .. and length                     [11]
        05  03E4    971         Rsb                                                 ; And return                        [11]
                    03E5    972
            00FC    03E5    973 .Entry BreakUp_File, ^M<R2, R3, R4, R5, R6, R7> ; Breakup_File routine               [11]
  54  04 AC    D0    03E7    974         MovL    04(Ap), R4                          ; Get length of filespec            [11]
        12  15    03EB    975         BLeq    10$                                 ; Skip ahead (exit)                 [11]
  55  08 AC    D0    03ED    976         MovL    08(Ap), R5                          ; Get address                       [11]
  56  0C AC    D0    03F1    977         MovL    12(Ap), R6                          ; Get address of file block         [11]
65  54  3A  3A    03F5    978         LocC    #^A":", R4, (R5)                    ; Scan for device part              [11]
        04  13    03F9    979         BEql    10$                                 ; Branch if none                    [11]
        57  D4    03FB    980         ClrL    R7                                  ; Index                             [11]
        D0  10    03FD    981         BsbB    OverLay                             ; Update pointers/etc.              [11]
        54  D5    03FF    982 10$:    TstL    R4                                  ; More String?                      [11]
        74  15    0401    983         BLeq    BreakUp_File_X                      ; If not, all done                  [11]
    50  65  9A    0403    984         MovZBL  (R5), R0                            ; Get next character                [11]
  5B 8F   50  91    0406    985         CmpB    R0, #^A"["                          ; If it's a directory start         [11]
        05  13    040A    986         BEql    20$                                 ; .. character, then                [11]
    3C  50  91    040C    987         CmpB    R0, #^A"<"                          ; .. use it                         [11]
        10  12    040F    988         BNeq    40$                                 ;                                   [11]
    50  02  80    0411    989 20$:    AddB2   #2, R0                              ; Form closing character            [11]
65  54  50  3A    0414    990         LocC    R0, R4, (R5)                        ; Is there a corresponding          [11]
        02  12    0418    991         BNeq    30$                                 ; .. closing char (']' or '>')?     [11]
        5B  11    041A    992         Brb     BreakUp_File_X                      ; If not, exit                      [11]
    57  01  D0    041C    993 30$:    MovL    #1, R7                              ; Setup index                       [11]
        AE  10    041F    994         BsbB    OverLay                             ; If it's OK, use it                [11]
65  54  2E  3A    0421    995 40$:    LocC    #^A".", R4, (R5)                    ; Is there a file type?             [11]
    55  51  D1    0425    996         CmpL    R1, R5                              ; If yes, and no name, go           [11]
        24  13    0428    997         BEql    60$                                 ;                                   [11]
        50  D5    042A    998         TstL    R0                                  ; Was it found at all?              [11]
```

```
                09  12  042C   999          BNeq    50$                             ; If so, get file name                  [11]
          65    54  3B  3A  042E  1000          LocC    #^A";", R4, (R5)                ; Was there a version?                  [11]
                55  51  D1  0432  1001          CmpL    R1, R5                          ; If it's right here,                   [11]
                    38  13  0435  1002          BEql    80$                             ; .. use it as filename term.           [11]
                        0437  1003  50$:                                               ; R1 points past end of name           [11]
          52    51  55  C3  0437  1004          SubL3   R5, R1, R2                      ; Get name length                       [11]
                53  55  D0  043B  1005          MovL    R5, R3                          ; Address of start of name              [11]
             10 A6  52  7D  043E  1006          MovQ    R2, 16(R6)                      ; Move to output                        [11]
                54  50  7D  0442  1007          MovQ    R0, R4                          ; Skip to terminator                    [11]
                    54  D5  0445  1008          TstL    R4                              ; Any file left?                        [11]
                    2E  15  0447  1009          BLeq    BreakUp_File_X                  ; If not, exit now                      [11]
                65  2E  91  0449  1010          CmpB    #^A".", (R5)                    ; Is there a type?                      [11]
                    21  12  044C  1011          BNeq    80$                             ; If not, check version                 [11]
                        044E  1012  60$:                                                ; Next character is '.'; type          [11]
                    54  D7  044E  1013          DecL    R4                              ; Skip past '.' to check for            [11]
                    25  19  0450  1014          BLss    BreakUp_File_X                  ; Exit if no more                       [11]
                    55  D6  0452  1015          IncL    R5                              ; .. '.' or ';' for version             [11]
          65    54  3B  3A  0454  1016          LocC    #^A";", R4, (R5)                ; Is there a version?                   [11]
                    04  12  0458  1017          BNeq    70$                             ; Yep, go off                           [11]
          65    54  2E  3A  045A  1018          LocC    #^A".", R4, (R5)                ; Alternate syntax for version          [11]
                        045E  1019                                                      ; R0, R1 at version or end             [11]
          52    51  55  C3  045E  1020  70$:    SubL3   R5, R1, R2                      ; Calculate length of                   [11]
                    52  D6  0462  1021          IncL    R2                              ; .. the file type string               [11]
          53    55  01  C3  0464  1022          SubL3   #1, R5, R3                      ; And it's address                      [11]
             18 A6  52  7D  0468  1023          MovQ    R2, 24(R6)                      ; And move it to output                 [11]
                54  50  7D  046C  1024          MovQ    R0, R4                          ; Update to terminator/end              [11]
                    54  D5  046F  1025  80$:    TstL    R4                              ; Is there more file?                   [11]
                    04  15  0471  1026          BLeq    BreakUp_File_X                  ; If not, exit                          [11]
          20 A6     54  7D  0473  1027          MovQ    R4, 32(R6)                      ; Use rest of string as version        [11]
                        0477  1028  BreakUp_File_X:                                    ; Exit routine                         [11]
             50     01  D0  0477  1029          MovL    #1, R0                          ; Set up status return                  [11]
                        04  047A  1030          Ret                                     ; And then return, of course           [11]
                            047B  1031
                            047B  1032          .END
```

```
$$ARGS          =  00000004     D
$$T1            =  00000014     D
ACTION             000000A1 R   D  03
BASE_LIST          0000027B R   D  03
BRANCH             00000099 R   D  03
BRANCH_IF_NONE     000000E6 R   D  03
BREAKUP_FILE       000003E5 RG  D  03
BREAKUP_FILE_X     00000477 R   D  03
CHECK_DIGIT        000002AE R   D  03
CHECK_RADIX        00000319 R   D  03
CHECK_SIGN         00000302 R   D  03
CLI$K_ALNUM     =  00000087     D
CLI$K_ALPHA     =  00000086     D
CLI$K_BIF       =  00000083     D
CLI$K_BIFS      =  00000094     D
CLI$K_BR        =  00000082     D
CLI$K_CALL      =  00000092     D
CLI$K_COMMA     =  0000008E     D
CLI$K_DEC       =  0000008A     D
CLI$K_EOL       =  00000091     D
CLI$K_ERROR     =  00000080     D
CLI$K_EXIT      =  00000081     D
CLI$K_FILE      =  00000095     D
CLI$K_HEX       =  00000089     D
CLI$K_KEYWORD   =  0000008C     D
CLI$K_NUM       =  00000085     D
CLI$K_OCT       =  00000088     D
CLI$K_RETURN    =  00000093     D
CLI$K_SLASH     =  00000090     D
CLI$K_SPACE     =  00000084     D
CLI$K_STRING    =  0000008B     D
CLI$K_SYMBOL    =  0000008D     D
CLI$K_VALUE     =  0000008F     D
DO_ACTION          000000E8 R   D  03
DS$GL_CLIBASE      ********     X  03
DS$GL_RADIX        ********     X  03
DS$GT_VALPHA       000001E6 RG  D  03
DS$GT_VALPHANUM    000001F8 RG  D  03
DS$GT_VFILE        0000020A RG  D  03
DS$GT_VHEX         0000026B RG  D  03
DS$GT_VSPACE       0000021C RG  D  03
DS$GT_VSYMBOL      000001D4 RG  D  03
DS$K_ERROR      =  00000002     D
DS$K_NORMAL     =  00000001     D
DS$K_SEVERE     =  00000004     D
DS$K_SUBSYS     =  00000066     D
DS$K_WARNING    =  00000000     D
DS$M_ABRTFLG    =  00000040     D
DS$M_BADTIME    =  00100000     D
DS$M_BATCH      =  00400000     D
DS$M_BRKCLR     =  00001000     D
DS$M_BRKPT      =  00000800     D
DS$M_CHARFLG    =  00000100     D
DS$M_CMDFLG     =  00000080     D
DS$M_CTRLC      =  00000001     D
DS$M_CTRLO      =  00010000     D
DS$M_DEVFLG     =  00000200     D
DS$M_DISABLCC   =  01000000     D
DS$M_DONFLG     =  00002000     D
DS$M_ERRFLG     =  00000010     D
DS$M_EXCEPT     =  00080000     D
DS$M_EXETST     =  00040000     D
DS$M_HLTFLG     =  00000008     D
DS$M_LODFLG     =  00000002     D
DS$M_MEMMGT     =  00008000     D
DS$M_OUTPUT     =  00800000     D
DS$M_RUBFLG     =  00000020     D
DS$M_SCRIPT     =  00200000     D
DS$M_SETIMR     =  02000000     D
DS$M_STRFLG     =  00000004     D
DS$M_SUBT       =  00004000     D
DS$M_SYSFLG     =  00000400     D
DS$M_TIMRON     =  00020000     D
DS$V_ABRTFLG    =  00000006     D
DS$V_BADTIME    =  00000014     D
DS$V_BATCH      =  00000016     D
DS$V_BRKCLR     =  0000000C     D
DS$V_BRKPT      =  0000000B     D
DS$V_CHARFLG    =  00000008     D
DS$V_CMDFLG     =  00000007     D
DS$V_CTRLC      =  00000000     D
DS$V_CTRLO      =  00000010     D
DS$V_DEVFLG     =  00000009     D
DS$V_DISABLCC   =  00000018     D
DS$V_DONFLG     =  0000000D     D
DS$V_ERRFLG     =  00000004     D
DS$V_EXCEPT     =  00000013     D
DS$V_EXETST     =  00000012     D
DS$V_HLTFLG     =  00000003     D
DS$V_LODFLG     =  00000001     D
DS$V_MEMMGT     =  0000000F     D
DS$V_OUTPUT     =  00000017     D
DS$V_RUBFLG     =  00000005     D
DS$V_SCRIPT     =  00000015     D
DS$V_SETIMR     =  00000019     D
DS$V_STRFLG     =  00000002     D
DS$V_SUBT       =  0000000E     D
DS$V_SYSFLG     =  0000000A     D
DS$V_TIMRON     =  00000011     D
DS$_ARITH       =  006600D0     D
DS$_ASBE        =  00660118     D
DS$_BADLINK     =  006600F0     D
DS$_BADTYPE     =  006600E8     D
DS$_BIIC        =  00660120     D
DS$_CHME        =  006600A8     D
DS$_CHMK        =  006600E0     D
DS$_DEVNAME     =  00660108     D
DS$_ERROR       =  00660002     D
DS$_FHWE        =  00660068     D
DS$_FRAGBUF     =  00660080     D
DS$_IBUSY       =  006600C8     D
DS$_IDERR       =  006600C0     D
DS$_IHWE        =  00660060     D
DS$_ILLCHAR     =  00660018     D
DS$_ILLPAGCNT   =  00660078     D
DS$_ILLUNIT     =  00660100     D
DS$_INSFMEM     =  00660050     D
DS$_IPL2HI      =  006600B8     D
DS$_IVADDR      =  00660040     D
DS$_IVVECT      =  00660038     D
DS$_KRNLSTK     =  00660090     D
DS$_LOGIC       =  00660070     D
DS$_MCHK        =  00660088     D
DS$_MMOFF       =  00660058     D
DS$_NEEDUNIT    =  006600F8     D
DS$_NODE        =  00660128     D
DS$_NOPCS       =  00660110     D
DS$_NORMAL      =  00660001     D
DS$_NOSUPPORT   =  006600B1     D
DS$_NOTDON      =  00660030     D
DS$_NOTIMP      =  006600B0     D
DS$_NULLSTR     =  00660010     D
DS$_OVERFLOW    =  00660008     D
DS$_POWER       =  00660098     D
DS$_PROGERR     =  00660020     D
DS$_SEVERE      =  00660004     D
DS$_TRANSL      =  006600A0     D
DS$_TRUNCATE    =  00660028     D
DS$_UNEXPINT    =  006600D8     D
DS$_VASFULL     =  00660048     D
DS$_WARNING     =  00660000     D
DSX$PARSE          00000007 RG  D  03
DSX$PARSE_X        00000068 R   D  03
DSX$SUPERPARSE     00000000 RG  D  03
EXIT               00000065 R   D  03
HP$A_DEPENDENT     00000032     D
HP$A_DEVICE        00000018     D
HP$A_DVA           0000001C     D
HP$A_LINK          00000020     D
HP$B_DRIVE         0000000B     D
HP$B_FLAGS         0000000A     D
HP$Q_DEVICE        00000000     D
HP$T_DEVICE        0000000C     D
HP$T_TYPE          00000026     D
HP$W_SIZE          00000008     D
HP$W_VECTOR        00000024     D
INVALID            000002FD R   D  03
OVERLAY            000003CF R   D  03
PARSE$_ACTION   =  0000000C     D
PARSE$_BUFADR   =  00000004     D
PARSE$_FLAG     =  00000010     D
PARSE$_NARGS    =  00000004     D
PARSE$_TREE     =  00000008     D
PARSE_COMMON       0000000B R   D  03
SAVE_CALL          00000000 R   D  03
SAVE_STATUS        00000004 R   D  03
SCAN$ALPHA         000001E4 RG  D  03
SCAN$ALPHANUM      000001F6 RG  D  03
SCAN$BINARY        00000296 RG  D  03
SCAN$COMPARE       00000249 RG  D  03
SCAN$DECIMAL       0000028C RG  D  03
```

```
SCAN$DEVICE          00000389 RG D  03
SCAN$FILE            00000208 RG D  03
SCAN$HEX             0000029B RG D  03
SCAN$NUMERIC         0000029E RG D  03
SCAN$OCTAL           00000291 RG D  03
SCAN$SEARCHLIST      0000033A RG D  03
SCAN$SPACE           0000021A RG D  03
SCAN$SPACES          0000021A RG D  03
SCAN$SPANC           0000022E RG D  03
SCAN$SYMBOL          000001D2 RG D  03
SIZ...           =   00000C01    D
SS$_NORMAL       =   00000001    D
TRAVERSE             0000001C R  D  03
TRVALNUM             00000112 R  D  03
TRVALPHA             0000010D R  D  03
TRVBIF               000000DE R  D  03
TRVBIFS              000000D2 R  D  03
TRVBR                00000097 R  D  03
TRVCALL              00000089 R  D  03
TRVCOMMA             0000016A R  D  03
TRVDEC               000000FC R  D  03
TRVEOL               000001B3 R  D  03
TRVERR               000000C7 R  D  03
TRVEXIT              00000063 R  D  03
TRVFILE              00000164 R  D  03
TRVHEX               000000F7 R  D  03
TRVKEYWORD           0000012F R  D  03
TRVNUM               00000101 R  D  03
TRVNUMA              00000108 R  D  03
TRVOCT               000000F2 R  D  03
TRVRETURN            00000069 R  D  03
TRVSLASH             0000016E R  D  03
TRVSPACE             000000ED R  D  03
TRVSTRING            00000117 R  D  03
TRVSYMBOL            00000160 R  D  03
TRVVALUE             00000173 R  D  03
TRV_COMMA_VALUE      00000176 R  D  03
V_BIT            =   00000002    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000032 (50.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000478 (1147.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...									
$$ARGS	=00000004	116	(1)	116	(1)								
$$T1	=00000C14	116	(1)	116	(1)								
ACTION	000000A1-R	282	(4)	#-238	(3)	#-249	(4)	#-270	(4)	#-297	(4)	#-306	(4)
				#-315	(4)	#-323	(5)	#-370	(5)	#-390	(6)		
BASE_LIST	0000027B-R	699	(11)	781	(12)								
BIT...	=00000003	117	(1)	114	(1)	115	(1)	117	(1)				
BRANCH	00000099-R	275	(4)	#-230	(3)	#-307	(4)	#-316	(4)	#-321	(5)	#-396	(6)
				#-479	(6)								
BRANCH_IF_NONE	000000E6-R	320	(5)	#-331	(5)	#-348	(5)	#-354	(5)	#-360	(5)	#-404	(6)
				#-410	(6)	#-464	(6)						
BREAKUP_FILE	000003E5-R	973	(16)										
BREAKUP_FILE_X	00000477-R	1028	(16)	#-1009	(16)	#-1014	(16)	#-1026	(16)	#-983	(16)	#-992	(16)
CHECK_DIGIT	000002AE-R	727	(12)										
CHECK_RADIX	00000319-R	776	(12)	#-723	(12)								
CHECK_SIGN	00000302-R	761	(12)	#-721	(12)	#-724	(12)						
CLI$K_ALNUM	=00000087	114	(1)										
CLI$K_ALPHA	=00000086	114	(1)										
CLI$K_BIF	=00000083	114	(1)										
CLI$K_BIFS	=00000094	114	(1)										
CLI$K_BR	=00000082	114	(1)										
CLI$K_CALL	=00000092	114	(1)										
CLI$K_COMMA	=0000008E	114	(1)										
CLI$K_DEC	=00000084	114	(1)										
CLI$K_EOL	=00000091	114	(1)										
CLI$K_ERROR	=00000080	114	(1)	#-224	(3)								
CLI$K_EXIT	=00000081	114	(1)										
CLI$K_FILE	=00000095	114	(1)										
CLI$K_HEX	=00000089	114	(1)										
CLI$K_KEYWORD	=0000008C	114	(1)										
CLI$K_NUM	=00000085	114	(1)										
CLI$K_OCT	=00000088	114	(1)										
CLI$K_RETURN	=00000093	114	(1)										
CLI$K_SLASH	=00000090	114	(1)										
CLI$K_SPACE	=00000084	114	(1)										
CLI$K_STRING	=0000008B	114	(1)										
CLI$K_SYMBOL	=0000008D	114	(1)										
CLI$K_VALUE	=0000008F	114	(1)										
DO_ACTION	000000E8-R	322	(5)	#-233	(3)	#-484	(6)						
DS$GL_CLIBASE	00000000-XR			286	(4)								
DS$GL_RADIX	00000000-XR			#-345	(5)								
DS$GT_VALPHA	000001E6-R	548	(8)										
DS$GT_VALPHANUM	000001F8-R	553	(8)										
DS$GT_VFILE	0000020A-R	558	(8)										
DS$GT_VHEX	0000026B-R	697	(11)	732	(12)								
DS$GT_VSPACE	0000021C-R	564	(8)										
DS$GT_VSYMBOL	000001D4-R	543	(8)										
DS$K_ERROR	=00000002	115	(1)										
DS$K_NORMAL	=00000001	115	(1)										
DS$K_SEVERE	=00000004	115	(1)										
DS$K_SUBSYS	=00000066	115	(1)	115	(1)								

```
DS$K_WARNING    =00000000      115     (1)
DS$M_ABRTFLG    =00000040      117     (1)
DS$M_BADTIME    =00100000      117     (1)
DS$M_BATCH      =00400000      117     (1)
DS$M_BRKCLR     =00001000      117     (1)
DS$M_BRKPT      =00000800      117     (1)
DS$M_CHARFLG    =00000100      117     (1)
DS$M_CMDFLG     =00000080      117     (1)
DS$M_CTRLC      =00000001      117     (1)
DS$M_CTRLO      =00010000      117     (1)
DS$M_DEVFLG     =00000200      117     (1)
DS$M_DISABLCC   =01000000      117     (1)
DS$M_DONFLG     =00002000      117     (1)
DS$M_ERRFLG     =00000010      117     (1)
DS$M_EXCEPT     =00080000      117     (1)
DS$M_EXETST     =00040000      117     (1)
DS$M_HLTFLG     =00000008      117     (1)
DS$M_LODFLG     =00000002      117     (1)
DS$M_MEMMGT     =00008000      117     (1)
DS$M_OUTPUT     =00800000      117     (1)
DS$M_RUBFLG     =00000020      117     (1)
DS$M_SCRIPT     =00200000      117     (1)
DS$M_SETIMR     =02000000      117     (1)
DS$M_STRFLG     =00000004      117     (1)
DS$M_SUBT       =00004000      117     (1)
DS$M_SYSFLG     =00000400      117     (1)
DS$M_TIMRON     =00020000      117     (1)
DS$V_ABRTFLG    =00000006      117     (1)
DS$V_BADTIME    =00000014      117     (1)
DS$V_BATCH      =00000016      117     (1)
DS$V_BRKCLR     =0000000C      117     (1)
DS$V_BRKPT      =0000000B      117     (1)
DS$V_CHARFLG    =00000008      117     (1)
DS$V_CMDFLG     =00000007      117     (1)
DS$V_CTRLC      =000C0000      117     (1)
DS$V_CTRLO      =00000010      117     (1)
DS$V_DEVFLG     =00000009      117     (1)
DS$V_DISABLCC   =00000018      117     (1)
DS$V_DONFLG     =0000000D      117     (1)
DS$V_ERRFLG     =00000004      117     (1)
DS$V_EXCEPT     =00000013      117     (1)
DS$V_EXETST     =00000012      117     (1)
DS$V_HLTFLG     =00000003      117     (1)
DS$V_LODFLG     =00000001      117     (1)
DS$V_MEMMGT     =0000000F      117     (1)
DS$V_OUTPUT     =00000017      117     (1)
DS$V_RUBFLG     =00000005      117     (1)
DS$V_SCRIPT     =00000015      117     (1)
DS$V_SETIMR     =00000019      117     (1)
DS$V_STRFLG     =00000002      117     (1)
DS$V_SUBT       =0000000E      117     (1)
DS$V_SYSFLG     =0000000A      117     (1)
DS$V_TIMRON     =00000011      117     (1)
DS$_ARITH       =006600D0      115     (1)
DS$_ASBE        =00660118      115     (1)
DS$_BADLINK     =006600F0      115     (1)
DS$_BADTYPE     =006600E8      115     (1)
```

```
DS$_BIIC          =00660120      115     (1)
DS$_CHME          =006600A8      115     (1)
DS$_CHMK          =006600E0      115     (1)
DS$_DEVNAME       =00660108      115     (1)
DS$_ERROR         =00660002      115     (1)     #-253     (4)     #-298     (4)
DS$_FHWE          =00660068      115     (1)
DS$_FRAGBUF       =00660080      115     (1)
DS$_ICBUSY        =006600C8      115     (1)
DS$_ICERR         =006600C0      115     (1)
DS$_IHWE          =00660060      115     (1)
DS$_ILLCHAR       =00660018      115     (1)
DS$_ILLPAGCNT     =00660078      115     (1)
DS$_ILLUNIT       =00660100      115     (1)
DS$_INSFMEM       =00660050      115     (1)
DS$_IPL2HI        =006600B8      115     (1)
DS$_IVADDR        =00660040      115     (1)
DS$_IVVECT        =00660038      115     (1)
DS$_KRNLSTK       =00660090      115     (1)
DS$_LOGIC         =00660070      115     (1)
DS$_MCHK          =00660088      115     (1)
DS$_MMOFF         =00660058      115     (1)
DS$_NEEDUNIT      =006600F8      115     (1)
DS$_NODE          =00660128      115     (1)
DS$_NOPCS         =00660110      115     (1)
DS$_NORMAL        =00660001      115     (1)
DS$_NOSUPPORT     =006600B1      115     (1)
DS$_NOTDON        =00660030      115     (1)
DS$_NOTIMP        =006600B0      115     (1)       115     (1)
DS$_NULLSTR       =0C360010      115     (1)
DS$_OVERFLOW      =00660008      115     (1)
DS$_POWER         =00660098      115     (1)
DS$_PROGERR       =00660020      115     (1)
DS$_SEVERE        =00660004      115     (1)
DS$_TRANSL        =006600A0      115     (1)
DS$_TRUNCATE      =00660028      115     (1)
DS$_UNEXPINT      =006600D8      115     (1)
DS$_VASFULL       =00660048      115     (1)
DS$_WARNING       =00660000      115     (1)       115     (1)
DSX$PARSE          00000007-R    187     (3)
DSX$PARSE_X        00000068-R    241     (3)     #-254     (4)     #-299     (4)
DSX$SUPERPARSE     00000000-R    184     (3)
EXIT               00000065-R    239     (3)     #-197     (3)     #-397     (6)
INVALID            000002FD-R    756     (12)
OVERLAY            000003CF-R    964     (16)    #-981     (16)    #-994     (16)
PARSE$_ACTION     =0000000C      116     (1)       288     (4)
PARSE$_BUFADR     =00000004      116     (1)     #-190     (3)
PARSE$_FLAG       =00000010      116     (1)
PARSE$_NARGS      =00000004      116     (1)
PARSE$_TREE       =00000008      116     (1)     #-193     (3)
PARSE_COMMON       0000000B-R    189     (3)     #-186     (3)
SAVE_CALL          00000000-R    126     (1)     #-192     (3)     #-251     (4)     #-264     (4)
SAVE_STATUS        00000004-R    128     (1)     #-250     (4)     #-263     (4)     #-307     (4)
SCAN$ALPHA         000001E4-R    546     (8)     #-353     (5)     #-920     (16)
SCAN$ALPHANUM      000001F6-R    551     (8)     #-359     (5)     #-910     (16)
SCAN$BINARY        00000296-R    710     (12)
SCAN$COMPARE       00000249-R    626     (10)    #-368     (5)
SCAN$DECIMAL       0000028C-R    702     (12)    #-922     (16)
```

```
SCAN$DEVICE       00000389-R   906    (16)
SCAN$FILE         00000208-R   556    (8)     #-409     (6)
SCAN$HEX          0000029B-R   714    (12)
SCAN$NUMERIC      0000029E-R   717    (12)    #-347     (5)     #-704     (12)    #-708     (12)    #-712     (12)
SCAN$OCTAL        00000291-R   706    (12)
SCAN$SEARCHLIST   0000033A-R   830    (14)    #-385     (6)
SCAN$SPACE        0000021A-R   562    (8)     #-330     (5)     #-439     (6)     #-459     (6)     #-472     (6)
SCAN$SPACES       0000021A-R   561    (8)
SCAN$SPANC        0000022E-R   571    (8)
SCAN$SYMBOL       000001D2-R   541    (8)     #-380     (6)     #-403     (6)
SIZ...          =00000001      117    (1)       117     (1)
SS$_NORMAL      =00000001      120    (1)     #-240     (3)
TRAVERSE          0000001C-R   194    (3)       198     (3)
TRVALNUM          00000112-R   358    (5)       224     (3)
TRVALPHA          000001DD-R   352    (5)       224     (3)
TRVBIF            000000DE-R   317    (4)       224     (3)
TRVBIFS           000000D2-R   305    (4)       224     (3)
TRVDR             00000097-R   269    (4)       224     (3)
TRVCALL           00000089-R   262    (4)       224     (3)
TRVCOMMA          0000016A-R   416    (6)       224     (3)
TRVDEC            000000FC-R   341    (5)       224     (3)
TRVEOL            00000163-R   471    (6)       224     (3)
TRVERR            000000C7-R   296    (4)       224     (3)
TRVEXIT           00000063-R   237    (3)       224     (3)
TRVFILE           00000164-R   408    (6)       224     (3)
TRVHEX            000000F7-R   338    (5)       224     (3)
TRVKEYWORD        0000012F-R   379    (6)       224     (3)
TRVNUM            00000101-R   344    (5)       224     (3)
TRVNUMA           00000108-R   346    (5)     #-337     (5)     #-340     (5)     #-343     (5)
TRVOCT            000000F2-R   335    (5)       224     (3)
TRVRETURN         00000069-R   248    (4)       224     (3)
TRVSLASH          0000016E-R   425    (6)       224     (3)
TRVSPACE          000000ED-R   329    (5)       224     (3)
TRVSTRING         00000117-R   361    (5)       224     (3)
TRVSYMBOL         00000160-R   402    (6)       224     (3)
TRVVALUE          00000173-R   434    (6)       224     (3)
TRV_COMMA_VALUE   00000176-R   437    (6)     #-418     (6)     #-427     (6)
V_BIT           =00000002      119    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	117 (1)	
$DEFINI	1	118 (1)	118 (1)
$DS_CLIDEF	1	114 (1)	114 (1)
$DS_DSDEF	2	115 (1)	115 (1)
$DS_HPODEF	2	118 (1)	118 (1)
$DS_PARSE_DEF	1	116 (1)	116 (1)
$EQU	1	117 (1)	114 (1) 115 (1)
$EQULS1	1	115 (1)	114 (1) 115 (1)
$EQULST	1	114 (1)	114 (1) 115 (1)
$GBLINI	2	114 (1)	114 (1) 115 (1) 117 (1)
$OFFDEF	1	116 (1)	116 (1)
$VIELD	1	117 (1)	117 (1)
$VIELD1	1	117 (1)	117 (1)
CASE	1	199 (3)	199 (3) 444 (6)
DSFDEF	3	117 (1)	117 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.12	00:00:00.31
Command processing	134	00:00:00.82	00:00:02.12
Pass 1	483	00:00:07.21	00:00:11.76
Symbol table sort	0	00:00:00.35	00:00:00.44
Pass 2	255	00:00:02.40	00:00:03.32
Symbol table output	19	00:00:00.15	00:00:00.17
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	34	00:00:00.75	00:00:00.84
Assembler run totals	969	00:00:11.84	00:00:19.00

The working set limit was 1000 pages.
36818 bytes (72 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 216 non-local and 53 local symbols.
1032 source lines were read in Pass 1, producing 0 object records in Pass 2.
45 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	4
DRB1:[DS.WORK]DS.MLB;218	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	12

237 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) PARSE/UPDA=(PARSE.UPD,PARSE.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

```
;    0001  0     %TITLE 'PCSLOAD Module'
;    0002  0     MODULE PCSLOAD (                        ! Dummy module for superviscrs
;    0003  0                     IDENT = '06-03'         ! that have no patch control store
;    0004  0                     ) =
;    0005  0     !++
;    0006  0     ! COPYRIGHT (c) 1980, 1981, 1983, 1984
;    0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0008  0     !
;    0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0014  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0015  0     ! REMAIN IN DEC.
;    0016  0     !
;    0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0018  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0019  0     ! CORPORATION.
;    0020  0     !
;    0021  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0023  0     !-
```

```
;    0024  0      !++
;    0025  0      ! Facility:
;    0026  0      !
;    0027  0      !       Diagnostic Supervisor
;    0028  0      !
;    0029  0      ! Abstract:
;    0030  0      !
;    0031  0      !       This module is used by those diagnostic supervisors that have
;    0032  0      !       no patch control store, namely, ESSAA, and ENSAA.
;    0033  0      !       The routines in this module do nothing except return status.
;    0034  0      !
;    0035  0      ! Author:
;    0036  0      !
;    0037  0      !       Bob Bergazzi
;    0038  0      !
;    0039  0      ! Creation Date:
;    0040  0      !
;    0041  0      !       22-Nov-1982
;    0042  0      !
;    0043  0      ! Version:
;    0044  0      !
;    0045  0      !       6.10
;    0046  0      !
;    0047  0      ! Modified By:
;    0048  0      !
;    0049  0      !       01      Bob Bergazzi    May 17, 1983    Version 6.11
;    0050  0      !               Changed the order of searching libraries.
;    0051  0      !
;    0052  0      !               02      Bob Bergazzi    Oct. 21, 1983   Version 6.13
;    0053  0      !                       Added INITPCS command routine, DSV$INITPCS.
;    0054  0      !
;    0055  0      !               03      Bob Bergazzi    Mar. 1, 1984    Version 7.0
;    0056  0      !                       Moved ECSAA functionality to PCSLOAD750.
;    0057  0      !--
```

```
;    0058  0     %SBTTL  'Libraries'
;    0059  1     BEGIN                           ! Begin the Module_Name module
;    0060  1
;    0061  1     LIBRARY
;    0062  1             '$Diag';                ! Use Diag.L32 first
;    0063  1
;    0064  1     LIBRARY
;    0065  1             '$DS';                  ! Use Ds.L32 second
;    0066  1
;    0067  1     LIBRARY
;    0068  1             '$CRD';                 ! Use CRD.L32 third
;    0069  1
;    0070  1     LIBRARY
;    0071  1             'Sys$Library:Lib';      ! Use Lib as last resort
```

```
;    0072  1     %SBTTL  'Table of Contents'
;    0073  1
;    0074  1     forward routine
;    0075  1         dsx$loadpcs,                    ! Interface to calling routine
;    0076  1         dsv$initpcs : novalue;          ! Init the PCS                          [02]
```

```
;    0077  1      %SBTTL  'Psect Definitions'
;    0078  1
;    0079  1      PSECT
;    0080  1          PLIT    = Data (NOEXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0081  1
;    0082  1      PSECT
;    0083  1          CODE    = Code (  EXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (WORD_RELATIVE));
;    0084  1
;    0085  1      PSECT
;    0086  1          OWN     = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0087  1
;    0088  1      PSECT
;    0089  1          GLOBAL  = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
```

```
;    0090  1     %SBTTL  'PCSLOAD-Static Storage'
;    0091  1
;    0092  1             ModNam ('PCSLOAD');             ! Module name for ErrSup macro
```

```
;    0093  1     %SBTTL 'DSX$LOADPCS Routine'
;    0094  1
;    0095  1     global! routine dsx$loadpcs =
;    0096  1
;    0097  1     !++
;    0098  1     ! Functional Description:
;    0099  1     !
;    0100  1     !       This routine does nothing but return status.
;    0101  1     !
;    0102  1     ! Formal Input Parameters:
;    0103  1     !
;    0104  1     !       None
;    0105  1     !
;    0106  1     ! Formal Output Parameters:
;    0107  1     !
;    0108  1     !       None
;    0109  1     !
;    0110  1     ! Side Effects:
;    0111  1     !
;    0112  1     !       None
;    0113  1     !
;    0114  1     ! Completion Codes:
;    0115  1     !
;    0116  1     !       ds$_nopcs       - PCS not available on this cpu.
;    0117  1     !--
```

```
;   0118  2     begin                                   ! routine dsx$loadpcs
;   0119  2     return ds$_nopcs;
;   0120  1     end;                                    ! routine dsx$loadpcs

                                                        .TITLE  PCSLOAD PCSLOAD Module
                                                        .IDENT  \06-03\

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                    44  41  4F  4C  53  43  50  07  00000 P.AAA:  .ASCII  <7>\PCSLOAD\                          ;
                                                        $MODULE=        P.AAA

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                            0000 00000          .ENTRY  DSX$LOADPCS, Save nothing               ; 0095
                                50 00660110   8F  D0 00002      MOVL    #6684944, R0                            ; 0119
                                                  04 00009      RET                                             ; 0120

; Routine Size:  10 bytes,    Routine Base:  CODE + 0000
```

```
;    0121  1     %SBTTL 'DSV$INITPCS Routine'                                !                        begin   [02]
;    0122  1
;    0123  1     global routine dsv$initpcs : novalue =
;    0124  1
;    0125  1     !++
;    0126  1     ! Functional Description:
;    0127  1     !
;    0128  1     !       This routine is called by CLI in response to an INITPCS command. The command is not valid for
;    0129  1     !       supervisors which use this module.
;    0130  1     !
;    0131  1     ! Formal Input Parameters:
;    0132  1     !
;    0133  1     !       None
;    0134  1     !
;    0135  1     ! Formal Output Parameters:
;    0136  1     !
;    0137  1     !       None
;    0138  1     !
;    0139  1     ! Side Effects:
;    0140  1     !
;    0141  1     !       None
;    0142  1     !
;    0143  1     ! Completion Codes:
;    0144  1     !
;    0145  1     !       None
;    0146  1     !--
```

```
;   0147  2     begin                                   ! routine dsv$initpcs
;   0148  2     $printi ($ascic (%string ('?? Command not valid for this supervisor!/')));
;   0149  1     end;                                    ! routine dsv$initpcs


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

74 6F 6E 20 64 6E 61 6D 6D 6F 43 20 3F 3F 2A  00008 P.AAB:  .ASCII  \*?? Command not valid for this superviso\  ;
73 69 68 74 20 72 6F 66 20 64 69 6C 61 76 20  00017                                                            ;
                 6F 73 69 76 72 65 70 75 73 20  00026                                                            ;
                                                        .ASCII  \r!/\                                           ;
                                                        .EXTRN  DSX$PRINTI
                                                        .PSECT  CODE,NOWRT,  SHR,2
                                                        .ENTRY  DSV$INITPCS, Save nothing                       ; 0123
                                        00000000'  EF   9F 00002        PUSHAB  P.AAB                           ; 0148
                              00000000G  9F             01   FB 00008  CALLS   #1, @#DSX$PRINTI                 ;
                                                        04   0000F    RET                                     ; 0149

; Routine Size:  16 bytes,    Routine Base:  CODE + 000A

;   0150  1
;   0151  1     end                                     ! end of pcsload module
;   0152  0     eludom
```

PSECT SUMMARY

Name	Bytes	Attributes
DATA	51	NOVEC,NOWRT, RD ,NOEXE, SHR, LCL, REL, CON,NOPIC,ALIGN(2)
CODE	26	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
DRB1:[DS.WORK]DIAG.L32;265	784	2	0	85	00:00.1
DRB1:[DS.WORK]DS.L32;159	653	2	0	42	00:00.2
DRB1:[DS.WORK]CRD.L32;265	483	0	0	62	00:00.2
SYS$SYSROOT:[SYSLIB]LIB.L32;7	18017	0	0	975	00:05.8

```
;                                       COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE PCSLOAD

; Size:         26 code + 51 data bytes
; Run Time:         00:08.4
; Elapsed Time:     00:28.5
; Lines/CPU Min:     1085
; Lexemes/CPU-Min:   2092
; Memory Used:  43 pages
; Compilation Complete
```

Table of contents

(1) 124 Libraries, Macros, and Equated Symbols
(1) 147 Work Psect Declarations
(1) 175 Data Psect Declarations
(1) 193 DSV$SetPage Routine - Set terminal page size
(1) 249 DSV$ShowPage Routine - Show terminal page size
(1) 295 DSX$Type_Out Routine - Do actual print
(1) 532 DSX$Print Routine - Load type code byte
(1) 608 DSX$PrintX Routines - Extended print routines
(1) 773 DSX$CvtReg Routine - Mnemonic register conversion routine

```
0000     1          .TITLE  PRINT   *** PRINT Formatted print routines
0000     2          .IDENT  /07-19/
0000     3          .NoShow Conditionals                ;                                   [10]
0000     4
0000     5 ;
0000     6 ; Copyright (c) 1977, 1983
0000     7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     8 ;
0000     9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    15 ; REMAIN IN DEC.
0000    16 ;
0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    19 ; CORPORATION.
0000    20 ;
0000    21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    23 ;--
```

```
                        0000     25 ;++
                        0000     26 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
                        0000     27 ;
                        0000     28 ; ABSTRACT:
                        0000     29 ;
                        0000     30 ; ENVIRONMENT:
                        0000     31 ;
                        0000     32 ; AUTHOR:       TOM SOUTTER     10-NOV-77       VERSION 01
                        0000     33 ;
                        0000     34 ; MODIFICATIONS:
                        0000     35 ;
                        0000     36 ;       01      Roger Riggs     12-Jun-78
                        0000     37 ;               Added DSX$PRINTI similar to DSX$PRINTF but always prints even
                        0000     38 ;               in APT mode. Changed prefix on print routines to DSX$PRINTx
                        0000     39 ;
                        0000     40 ;       02      Roger Riggs     12-Jun-78
                        0000     41 ;               Added usermode cancel and setmode
                        0000     42 ;
                        0000     43 ;       03      Roger Riggs     1-OCT-1979
                        0000     44 ;               Changed W^ to L^ on DSA$GL_FLAGS references
                        0000     45 ;
                        0000     46 ;       04      Roger Riggs, 14-Jan-1979, Version 5.2
                        0000     47 ;               Added code to DSX$CVTREG to allow the user
                        0000     48 ;               to specify a code for the mnemonic string instead of
                        0000     49 ;               the string itself codes are used for transportabile
                        0000     50 ;               diagnostics and represent a generic type of data,
                        0000     51 ;               i.e. UNIBUS map register.
                        0000     52 ;
                        0000     53 ;       05      Dave Butenhof 29-apr-80
                        0000     54 ;               Use RMS for output if DS$V_BATCH flag is set (running
                        0000     55 ;               under VMS batch/command file).
                        0000     56 ;
                        0000     57 ;       06      Dave Butenhof, 16-oct-1980, version 6.1
                        0000     58 ;               Set flag DS$V_OUTPUT on PRINT call.
                        0000     59 ;               Also have print use stack buffer for FAO output to avoid
                        0000     60 ;               conflict with SCRIPT prompt formatting.
                        0000     61 ;
                        0000     62 ;       07      - Jack Stansbury, 28-Oct-1981, Version 6.-
                        0000     63 ;               Added .LIBRARY statements for $DS and $DIAG. Fixed
                        0000     64 ;               truncation errors.
                        0000     65 ;
                        0000     66 ;       08      - Dave Butenhof, 02-Nov-1981, version 6.-
                        0000     67 ;               Fix truncation error.
                        0000     68 ;
                        0000     69 ;       09      - Dave Butenhof, 16-Nov-1981, version 6.5
                        0000     70 ;               Enhance to provide capability of outputting a binary 'type
                        0000     71 ;               code' byte preceding each and every message output by the
                        0000     72 ;               Supervisor.  This is controlled by the SET [CLEAR] BINARY
                        0000     73 ;               command.
                        0000     74 ;
                        0000     75 ;       10      - Jack Stansbury, 23-Mar-1982, Version 6.?
                        0000     76 ;               Took out the RTypeMsg routine because it is now obselete
                        0000     77 ;               with typecoding and with the Print routine. Also added a
                        0000     78 ;               "!/" to the message output by the print routine when
                        0000     79 ;               Debug_The_Sucker has been set to one.
                        0000     80 ;
                        0000     81 ;       11      Jack Stansbury, 8-Apr-1982, Version 6.7
```

```
                                ;               Added code to the DSX$TypeOut routine to call the CRD
                                ;               routine. Also added a .Library statement for /$CRD/,
                                ;               the new CRD library. Also changed some statements to
                                ;               the new DS macros (Br_If_ ...).
                                ;
                                ;       12      Marion Baggett, 9-Apr-1982, Version 6.7
                                ;               Ordered .library statements.
                                ;
                                ;       13      Jack Stansbury, 12-Apr-1982, Version 6.7
                                ;               Changed the code in DSX$TypeOut a bit so that the CRD routine
                                ;               is not called if the TypeCode is CRD_AutoTest. This allows CRD
                                ;               Print's to go through without the CRD routine being called.
                                ;
                                ;       14      Jack Stansbury, 6-May-1982, Version 6.8
                                ;               Added code in DSX$TypeOut to supress outputting the binary
                                ;               typecode when CRD-AutoTest is running (which, in itself, implies
                                ;               that the BINARY flag has been set on).
                                ;
                                ;       15      Jack Stansbury, 24-July-1982, Version 6.9
                                ;               Added support for the Menu Test of CRD.
                                ;
                                ;       16      Bob Bergazzi    30-March-1983   Version 6.11
                                ;               Added guts of SET PAGE command: DSV$SHOWPAGE, DSV$SETPAGE, and
                                ;               in DSX$TYPE_OUT.
                                ;
                                ;       17      John Ciukaj,    2-Apr.-1983,    Version 6.11
                                ;               - Changed references to CRD bits in DSA Flags since
                                ;                 they were redefined to distinguish between online
                                ;                 and offline.
                                ;
                                ;       18      Peter Green     18-JUL-1983     Version 6.12
                                ;               Changed print macros to test R0 for status such that
                                ;               on error an immediate exit occurs instead of attempting
                                ;               to print the message.
                                ;
                                ;       19      Richard Brown   7-June-1984     Version 7.0
                                ;               Added null arguments to SYS$QIO call in DSX$TYPE_OUT.
                                ;               Changed channel number from TTIN to TTOUT in two $QIO
                                ;               calls in DSX$TYPE_OUT, prompting for user response
                                ;               to display next page of terminal output.
                                ;--
```

```
0000    124          .Subtitle       Libraries, Macros, and Equated Symbols
0000    125 ;
0000    126 ; INCLUDE FILES:
0000    127 ;
0000    128          .Library        /Sys$Library:Lib/       ;                                [12]
0000    129          .Library        /$CRD/                  ; The CRD library                [11]
0000    130          .Library        /$DS/                   ;
0000    131          .Library        /$Diag/                 ;
0000    132
0000    133 ;
0000    134 ; EQUATED SYMBOLS:
0000    135 ;
0000    136
0000    137          $SSDEF                                  ;                                [18]
0000    138          $DS_DSDEF                               ; DIAGNOSTIC SUPERVISOR ERROR CODES
0000    139          $DS_DSADEF                              ; CONTROL FLAG DEFINITIONS
0000    140          $DS_CVTDEF                              ; Define CVTREG code values
0000    141          DSFDEF
0000    142          $RABDEF                                 ; Define RAB offsets
0000    143          $DS_TypeDef                             ; Define the TypeCode literals   [13]
0000    144          $CRD_Literals                           ; Define the CRD literals        [11]
0000    145          CLIDEF                                  ; Define CLI offsets             [16]
```

```
                                        0000   147          .SubTitle       Work Psect Declarations
                                    00000000   148          .Psect  Work, Noshr, Noexe, Wrt, Long
                                        0000   149
                                        0000   150 ;
                                        0000   151 ; OWN STORAGE:
                                        0000   152 ;
                                        0000   153
                                0000 0000'  0000   154 QBUFF:  .WORD   DS$K_BUFSIZ,0           ; QUADWORD BUFFER DESCRIPTOR
                                 00000000'  0004   155          .LONG   DS$GT_BUFFER
                                        0008   156
                                      00   0008   157 Width_Too_Large:        .byte   0       ; Bit 0 is set if the last line was     [16]
                                        0009   158                                            ; longer than the terminal width        [16]
                                 00000000   0009   159 Line_Count::    .long   0               ; current number of lines output since  [16]
                                        000D   160                                            ; last PAGE                             [16]
                                 00000000   000D   161 DS$GL_Page::    .long   0               ; SET PAGE 0 is default                 [16]
                                        0011   162 T_Page:                                     ;                                       [16]
6D 20 65 7A 69 73 20 65 67 61 50 00'    0011   163          .ASCIC  "Page size must be greater than 0!/"    ;                       [16]
74 61 65 72 67 20 65 62 20 74 73 75     001D
   2F 21 30 20 6E 61 68 74 20 72 65     0029
                                 22     0011
                                        0034   164 T_Show_Page:                                ;                                       [16]
61 70 20 6C 61 6E 69 6D 72 65 54 00'    0034   165          .ASCIC  "Terminal page size is !ZB line!%S!/"   ;                       [16]
21 20 73 69 20 65 7A 69 73 20 65 67     0040
2F 21 53 25 21 65 6E 69 6C 20 42 5A     004C
                                 23     0034
                                0000000D  0058   166 CR=13
                                0000000A  0058   167 LF=10
55 54 45 52 20 73 73 65 72 50 0A 0D     0058   168 Page_String:    .ascii  <CR><LF> -      ;                                       [16]
2C 65 72 6F 6D 20 72 6F 66 20 4E 52     0064
74 20 43 2D 6C 6F 72 74 6E 6F 43 20     0070
          2E 2E 2E 74 69 78 65 20 6F    007C
                                        0085   169                             /Press RETURN for more, Control-C to exit.../   ;       [16]
                                        0085   170
                                        0085   171 DS$GB_VDS_Debug::                          ; Make more contemporary.               [15]
                                        0085   172 Debug_The_Sucker::                         ; Print typecode number if = 1          [15]
                                     00 0085   173          .Byte   0                       ; Initial 0                             [15]
```

```
                     0086   175          .SubTitle       Data Psect Declarations
                 00000000   176          .Psect  DATA, SHR, NOEXE, NOWRT, BYTE
                     0000   177
                     0000   178 ;+
                     0000   179 ; This address list is indexed by (-CODE-1) passed to DS$CVTREG as MNEADR
                     0000   180 ; Each address points to data in ONLY??? lists of the form:
                     0000   181 ;       CVT$T_....:
                     0000   182 ;                .ADDRESS        10$     Address of ASCIC string
                     0000   183 ;                .LONG   V1,...V6        Parameters V1-V6, only if required
                     0000   184 ;       10$:     .ASCIC  "mnemonic string"
                     0000   185 ;-
                     0000   186
                     0000   187 CVT$A_CODELIST:
          00000000'  0000   188          .ADDRESS        CVT$T_UBADPR    ; Args for UNIBUS datapath CVTREG
          00000000'  0004   189          .ADDRESS        CVT$T_UBAMAP    ; Args for UNIBUS MAP CVTREG
          00000000'  0008   190          .ADDRESS        CVT$T_MBASR     ; Args for MASSBUS Status CVTREG
          00000000'  000C   191          .ADDRESS        CVT$T_MBAMAP    ; Args for MASSBUS MAP CVTREG
```

```
                     0010   193          .Subtitle       DSV$SetPage Routine - Set terminal page size
                 00000000   194          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0000   195 ;++                                                                          [16]
                     0000   196 ; Functional Description:                                                  [16]
                     0000   197 ;                                                                          [16]
                     0000   198 ;       This is a CLI command routine.  It loads the global DS$GL_PAGE with [16]
                     0000   199 ;       a value specified to CLI.  If the page size is out of range, an error [16]
                     0000   200 ;       message is typed and it returns a failure status code (though CLI  [16]
                     0000   201 ;       does not currently recognize return status).                       [16]
                     0000   202 ;                                                                          [16]
                     0000   203 ; Calling sequence:                                                       [16]
                     0000   204 ;                                                                          [16]
                     0000   205 ;       JSB     DSV$SETPAGE                                               [16]
                     0000   206 ;                                                                          [16]
                     0000   207 ; Input Parameters:                                                       [16]
                     0000   208 ;                                                                          [16]
                     0000   209 ;       none                                                              [16]
                     0000   210 ;                                                                          [16]
                     0000   211 ; Output Parameters:                                                      [16]
                     0000   212 ;                                                                          [16]
                     0000   213 ;       none                                                              [16]
                     0000   214 ;                                                                          [16]
                     0000   215 ; Implicit Inputs:                                                       [16]
                     0000   216 ;                                                                          [16]
                     0000   217 ;       R2 points to CLI data table                                       [16]
                     0000   218 ;                                                                          [16]
                     0000   219 ; Implicit Outputs:                                                      [16]
                     0000   220 ;                                                                          [16]
                     0000   221 ;       none                                                              [16]
                     0000   222 ;                                                                          [16]
                     0000   223 ; Side effects:                                                          [16]
                     0000   224 ;                                                                          [16]
                     0000   225 ;       Changes the value of DS$GL_PAGE variable                          [16]
                     0000   226 ;                                                                          [16]
                     0000   227 ; Status returns:                                                        [16]
                     0000   228 ;                                                                          [16]
                     0000   229 ;       0       failure (value out of range)                              [16]
                     0000   230 ;       1       success                                                   [16]
                     0000   231 ;--                                                                       [16]
```

```
                                0000    233 DSV$SetPage::                                         ;                                    [16]
              50    1C A2   D0  0000    234             MOVL    CLI$L_DATA(R2), R0                ; Get the page size                  [16]
                       0B   19  0004    235             BLSS    10$                               ; Too small?                         [16]
                                0006    236                                                                                            
      0000000D'EF      50   D0  0006    237             MOVL    R0, DS$GL_PAGE                    ; No, store the value                [16]
                                000D    238                                                                                            
                 50    01   D0  000D    239             MOVL    #1, R0                            ; Success                            [16]
                            05  0010    240             RSB                                       ;                                    [16]
                                0011    241                                                                                            
                                0011    242 10$:        $Print  -                                 ; Page size out of range             [16]
                                0011    243                     #DS$K_Type_Command_Err, -                ; typecode                    [16]
                                0011    244                     #DS$K_PrintF, -                   ;                                    [16]
                                0011    245                     T_Page                            ; the message                        [16]
                       50   D4  0024    246             CLRL    R0                                ; Failure                            [16]
                            05  0026    247             RSB                                       ;                                    [16]
```

```
                        0027   249          .Subtitle       DSV$ShowPage Routine - Show terminal page size
                        0027   250 ;++                                                                          [16]
                        0027   251 ; Functional Description:                                                  [16]
                        0027   252 ;                                                                           [16]
                        0027   253 ;       This is a CLI command routine.  It prints the current DS$GL_PAGE.  [16]
                        0027   254 ;                                                                           [16]
                        0027   255 ; Calling sequence:                                                       [16]
                        0027   256 ;                                                                           [16]
                        0027   257 ;       JSB     DSV$SHOWPAGE                                              [16]
                        0027   258 ;                                                                           [16]
                        0027   259 ; Input Parameters:                                                       [16]
                        0027   260 ;                                                                           [16]
                        0027   261 ;       none                                                              [16]
                        0027   262 ;                                                                           [16]
                        0027   263 ; Output Parameters:                                                      [16]
                        0027   264 ;                                                                           [16]
                        0027   265 ;       none                                                              [16]
                        0027   266 ;                                                                           [16]
                        0027   267 ; Implicit Inputs:                                                       [16]
                        0027   268 ;                                                                           [16]
                        0027   269 ;       none                                                              [16]
                        0027   270 ;                                                                           [16]
                        0027   271 ; Implicit Outputs:                                                      [16]
                        0027   272 ;                                                                           [16]
                        0027   273 ;       none                                                              [16]
                        0027   274 ;                                                                           [16]
                        0027   275 ; Side effects:                                                          [16]
                        0027   276 ;                                                                           [16]
                        0027   277 ;       none                                                              [16]
                        0027   278 ;                                                                           [16]
                        0027   279 ; Status returns:                                                        [16]
                        0027   280 ;                                                                           [16]
                        0027   281 ;       1       success                                                   [16]
                        0027   282 ;--                                                                        [16]
```

```
                                0027    284 DSV$ShowPage::                                  ;                                                       [16]
          50    0000000D'EF     9A  0027    285         MOVZBL  L^DS$GL_Page, R0            ; Get the current page size                             [16]
                                    002E    286         $Print  -                           ; Print with typecode                                   [16]
                                    002E    287                 #DS$K_Type_Command_Out, -       ; typecode                                      [16]
                                    002E    288                 #DS$K_PrintF, -             ; the type of PRINT routine                             [16]
                                    002E    289                 T_Show_Page, -              ; the message                                           [16]
                                    002E    290                 R0                          ; the page size                                         [16]
                50  01      D0  0043    291         MOVL    #1, R0                          ; Success                                               [16]
                                05  0046    292         RSB                                 ;                                                       [16]
                                    0047    293
```

```
                     0047   295          .Subtitle       DSX$Type_Out Routine - Do actual print
                  00000047   296          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0047   297 ;++
                     0047   298 ; Functional Description:
                     0047   299 ;
                     0047   300 ;       This routine does the actual work of outputting.  If the flag
                     0047   301 ;       DSA$V_BATCH is set, then it will use RMS to output to a file
                     0047   302 ;       (presumably) otherwise it will use QIO.  This routine also does          [16]
                     0047   303 ;       the work for SET PAGE online and standalone by traipsing through         [16]
                     0047   304 ;       the ouput buffer and comparing the number of linefeeds to the value      [16]
                     0047   305 ;       specified by the most recent SET PAGE command.  The routine also must    [16]
                     0047   306 ;       predict where the VDS and VMS will insert a CRLF due to the line being   [16]
                     0047   307 ;       too long for the current terminal width.;
                     0047   308 ;
                     0047   309 ; Input Parameters:
                     0047   310 ;
                     0047   311 ;       4(ap)           Length of string to output
                     0047   312 ;       8(ap)           Address of string
                     0047   313 ;
                     0047   314 ; Output Parameters:
                     0047   315 ;
                     0047   316 ;       none
                     0047   317 ;
                     0047   318 ; Implicit Inputs:
                     0047   319 ;
                     0047   320 ;       DS$GB_WIDTH     terminal width
                     0047   321 ;       DS$GL_PAGE      terminal page size
                     0047   322 ;
                     0047   323 ; Implicit Outputs:
                     0047   324 ;
                     0047   325 ;       none
                     0047   326 ;
                     0047   327 ; Side effects:
                     0047   328 ;
                     0047   329 ;       If either the CRD_AutoTest bit or the CRD_MenuTest bit is set (in the    [15]
                     0047   330 ;       DSA flags), this routine will call a CRD routine to determine if the     [15]
                     0047   331 ;       output should actually be printed. If the CRD routine returns success,  [15]
                     0047   332 ;       do NOT print.                                                            [15]
                     0047   333 ;
                     0047   334 ; Status returns:
                     0047   335 ;
                     0047   336 ;       Any code from QIOW or PUT
                     0047   337 ;--
```

```
                                00FC  0047   339 .Entry DSX$Type_Out, ^M<R2,R3,R4,R5,R6,R7>              ;                [16]
                                      0049   340
                00000000'EF     1A    91  0049   341         CmpB    #DS$K_Type_CRD_AutoTest, ~ ; Compare the current typecode to ...[13]
                                      0050   342                   L^DS$GB_TypeCode          ; ... to CRD_AutoTest. The same  [13]
                                2D    13  0050   343         Beql    50$                       ; If same, branch to print it.   [13]
                                      0052   344                                             ; If not same, continue ...      [13]
                                      0052   345
                                      0052   346         Br_If_CRD_AutoTest_Off  10$       ; If under Auto Test, then branch [17]
                                      005A   347         Br_If_CRD_MenuTest_Off  10$       ; If under Menu Test, then branch [17]
                                1B    11  0062   348         Brb     50$                       ; Else, not under either one.    [15]
                                      0064   349
                          08 AC     DD  0064   350 10$:     PushL   8(AP)                     ; Push Address as param-4        [15]
                       7E 04 AC     3C  0067   351         MovZWL  4(AP), -(SP)              ; Push Length as param-3         [11]
                7E  00000000'EF     9A  006B   352         MovZBL  L^DS$GB_TypeCode, -(SP)   ; Push TypeCode as param-2       [11]
                                01    DD  0072   353         PushL   #CRD$K_TypeCode           ; Push "type of call" as param-1 [11]
                00000000'FF     04    FB  0074   354         CallS   #4, @L^DS$GA_CRD_DS_Interface   ; Call indirectly the ... [11]
                                      007B   355                                             ; ... CRD$DS_Interface routine.  [11]
                          01 50     E9  007B   356         BlbC    R0, 50$                   ; Print if the routine returns failure [11]
                                04    007E   357         Ret                               ; Return if routine returns success [11]
                                      007F   358
                                      007F   359 50$:     Br_If_Not_Batch 100$              ; Branch if use QIO for output   [11]
                                      0087   360
                                      0087   361         ;+
                                      0087   362         ; We are running in Batch mode. That is, we are using RMS to output [14]
                                      0087   363         ; everything. Test to see if either CRD Auto Test or Menu Test is   [15]
                                      0087   364         ; executing. If so, assume the BINARY flag has been set and remove the [15]
                                      0087   365         ; binary typecode from the start of the string.                     [15]
                                      0087   366         ;-
                                      0087   367
      50    00000000'EF     9E  0087   368         MovAB   L^DS$Rab_Output, R0       ; Point to RAB                   [09]
            51    04 AC     3C  008E   369         MovZWL  4(AP), R1                 ; Length to R1                   [14]
            52    08 AC     D0  0092   370         MovL    8(AP), R2                 ; Address to R2                  [14]
                                      0096   371
                                      0096   372         Br_If_CRD_AutoTest_Off  60$       ; If running under Auto Test, branch [17]
                                      009E   373         Br_If_CRD_MenuTest_Off  60$       ; If running under Menu Test, branch [17]
                                04    11  00A6   374         Brb     70$                       ; Not either one, so skip over it [15]
                                      00A8   375
                                51    D7  00A8   376 60$:     DecL    R1                        ; Length = Length - 1            [15]
                                52    D6  00AA   377         IncL    R2                        ; Address = Address + 1 (byte)   [14]
                                      00AC   378
                          22 A0     51    B0  00AC   379 70$:     MovW    R1, B^Rab$W_Rsz(R0)       ; Get record length              [14]
                          28 A0     52    D0  00B0   380         MovL    R2, B^Rab$L_Rbf(R0)       ; Get record address             [14]
                                      00B4   381         $Put    Rab=(R0)                  ; Output string                  [09]
                                04    00BD   382         Ret                               ; Go to caller.                  [09]
                                      00BE   383
                                      00BE   384         ;+
                                      00BE   385         ; We are not running in Batch mode - we are using QIO to output [14]
                                      00BE   386         ; everything. Perform the same check as the above performs for CRD. [14]
                                      00BE   387         ;-
                                      00BE   388
        03 00000000'EF        E9    00BE   389 100$:     blbc    L^ds$gb_qioact, 200$      ; Branch if QIO not active       [09]
                      0386    30    00C5   390         bsbw    reset_input               ; Reset input channel if active  [09]
                                      00C8   391
                                7E    7C  00C8   392 200$:     clrq    -(sp)                     ; p5/p6 zero                     [09]
                                7E    7C  00CA   393         clrq    -(sp)                     ; p3/p4 zero                     [09]
                                      00CC   394
            51    04 AC     3C    00CC   395         MovZWL  4(AP), R1                 ; Move length to R1              [14]
```

```
                      52    08 AC    D0  00D0    396          MovL    8(AP), R2                   ; Move address to R2                                [14]
                                      00D4    397
                                      00D4    398          Br_If_CRD_AutoTest_Off  250$    ; If running under Auto Test, branch    [17]
                                      00DC    399          Br_If_CRD_MenuTest_Off  250$    ; If running under Menu Test, branch    [17]
                            04    11  00E4    400          Brb     300$                        ; Not either one, so skip over it       [15]
                                      00E6    401
                            51    D7  00E6    402 250$:    DecL    R1                          ; Length = Length - 1                   [15]
                            52    D6  00E8    403          IncL    R2                          ; Address = Address + 1 (byte)          [14]
                                      00EA    404
                         55   51    D0  00EA    405 300$:    MovL    R1, R5                      ; R5 gets length                        [16]
                              54    D4  00ED    406          ClrL    R4                          ; Number of characters in current line  [16]
                         56   55    D0  00EF    407 310$:    Movl    R5, R6                      ; Save original length of string        [16]
                         57   52    D0  00F2    408          Movl    R2, R7                      ; Save original address of string       [16]
                   0000000D'EF    D5  00F5    409          TstL    DS$GL_Page                  ; SET PAGE 0 ?                          [16]
                              73    13  00FB    410          BEql    370$                        ; Yes, go right to QIOW                 [16]
                                      00FD    411
    0000000D'EF    00000009'EF    D1  00FD    412          CmpL    Line_Count, DS$GL_Page      ; Are we at page limit?                 [16]
                              03    19  0108    413          BLss    320$                        ; No                                    [16]
                            0087    31  010A    414          BrW     380$                        ; Yes, prompt user                      [16]
                                      010D    415
                53    00000000'EF    9A  010D    416 320$:    MovZBL  DS$GB_Width, R3             ; Get terminal width                    [16]
                      00000008'EF    94  0114    417          ClrB    Width_Too_Large             ; Clear flag indicating line too long   [16]
                                      011A    418
                              55    D7  011A    419 330$:    DecL    R5                          ; Any characters left?                  [16]
                              4C    19  011C    420          BLss    355$                        ; No, we have less than a page of chars [16]
                                      011E    421
                         62   0A    91  011E    422          CmpB    #^X0A, (R2)                 ; LF ?                                  [16]
                              37    13  0121    423          BEql    340$                        ; Yes                                   [16]
                                      0123    424
                         82   0D    91  0123    425          CmpB    #^X0D, (R2)+                ; No, CR ?                              [16]
                              04    12  0126    426          BNEq    331$                        ; No                                    [16]
                              54    D4  0128    427          ClrL    R4                          ; Yes, zero width count                 [16]
                              EE    11  012A    428          BrB     330$                        ; Continue with next character          [16]
                                      012C    429
                      FF A2   09    91  012C    430 331$:    CmpB    #^X09, -1(R2)               ; No, TAB ?                             [16]
                              09    12  0130    431          BNEq    335$                        ; No                                    [16]
                              54    D6  0132    432 333$:    IncL    R4                          ; Yes, add 1 to width                   [16]
                         54   07    93  0134    433          BitB    #7, R4                      ; Are we at TAB boundary?               [16]
                              F9    12  0137    434          BNEq    333$                        ; No, count another                     [16]
                              54    D7  0139    435          DecL    R4                          ; Yes, we added one to many so subtract [16]
                      DB 54   53    F2  013B    436 335$:    AOBLss  R3, R4, 330$                ; Width gets larger by 1                [16]
                                      013F    437          Br_If_Not_User  337$            ; S/A doesn't pad next line if TAB      [16]
                         54   53    C2  0147    438          SubL2   R3, R4                      ; How far over the edge of width?       [16]
                              12    12  014A    439          BNEq    353$                        ; Too far, don't do extra CRLF, save R4 [16]
                                      014C    440                                              ; We are at the edge of the width,      [16]
                         62   0D    91  014C    441 337$:    CmpB    #^X0D, (R2)                 ; If CR is next, line is not too long.  [16]
                              C9    13  014F    442          BEql    330$                        ; so print CRLF also                    [16]
                00000008'EF   01    88  0151    443          BiSB    #1, Width_Too_Large         ; No CRLF there, we need to do an extra [16]
                                      0158    444                                              ; CRLF if we are also at the page limit [16]
                              02    11  0158    445          BrB     350$                        ; Count another line                    [16]
                                      015A    446
                              52    D6  015A    447 340$:    IncL    R2                          ; Point to next character               [16]
                              54    D4  015C    448 350$:    ClrL    R4                          ; Clear width count                     [16]
A3 00000009'EF    0000000D'EF   F2    015E    449 353$:    AOBLss  DS$GL_Page, Line_Count, 320$    ; Enough for a page?                    [16]
                                      016A    450
                              54    D4  016A    451 355$:    ClrL    R4                          ; Clear width count                     [16]
                         55   52    57  016C    452          SubL3   R7, R2, R5                  ; R5 gets length of this string         [16]
```

```
                        0170   453
                        0170   454 370$:
                     7E 7C  0170   455          CLRQ    -(SP)                   ; Two null arguments                [19]
                     7E 7C  0172   456          CLRQ    -(SP)                   ; Two null arguments                [19]
                     55 DD  0174   457          PushL   R5                      ; Push length of string                     [16]
                     57 DD  0176   458          PushL   R7                      ; Push address of string                    [16]
                            0178   459
                     7E 7C  0178   460          clrq    -(sp)                   ; astadr/astprm ..ero                       [14]
                     7E D4  017A   461          clrl    -(sp)                   ; iosb zero                                 [09]
                     00' DD 017C   462          pushl   S^#io$_writevblk        ; function                                  [09]
          00000000'EF    DD 017E   463          pushl   L^ds$gw_ttout           ; channel                                   [09]
                     7E D4  0184   464          clrl    -(sp)                   ; event flag                                [09]
00000000'GF          0C FB 0186   465          calls   #12, G^sys$qiow         ; do the qio                                [09]
                            018D   466
                  55 56 55 C3  018D   467          SubL3   R5, R6, R5              ; R1 gets number of characters left         [16]
                     01 14  0191   468          BGtr    380$                    ; There's more, prompt user                 [16]
                        04  0193   469          Ret                             ; None left                                 [16]
                            0194   470
          00000008'EF    01 93  0194   471 380$:   BiTB    #1, Width_Too_Large     ; Was the last line too long?               [16]
                     29 13  0198   472          BEql    385$                    ; No                                        [16]
                            019D   473          $QIOW_S , -                     ; Yes, do an extra CRLF before prompt       [16]
                            019D   474                  L^DS$GW_TTOUT, -        ; Channel number                            [16]
                            019D   475                  S^#IO$_WRITEVBLK, -     ; Function                                  [16]
                            019D   476                  P1 = Page_String, -     ; Output buffer address                     [16]
                            019D   477                  P2 = #2                 ; Output buffer size                        [16]
          00000008'EF    94 01C0   478          ClrB    Width_Too_Large         ; Reset width flag                          [16]
                            01C6   479
                            01C6   480 385$:   $QIOW_S , -                     ; Prompt for more output                    [16]
                            01C6   481                  L^DS$GW_TTOUT, -        ; Channel number                            [16]
                            01C6   482                  #IO$_WRITEVBLK, -       ; Function code                             [16]
                            01C6   483                  P1 = Page_String, -     ; Output buffer address                     [16]
                            01C6   484                  P2 = #45                ; Output buffer size                        [16]
                     7E 94  01EB   485          ClrB    -(SP)                   ; Space for input buffer                    [16]
                  57 6E DE  01ED   486          MovAL   (SP), R7                ; Input buffer on stack                     [16]
                            01F0   487 387$:   Br_If_Not_User  388$            ;                                           [16]
          00000000'EF    01 88  01F8   488          BiSB    #1, DS$GB_QIOACT        ; Set input QIO active                      [16]
                            01FF   489                                         ; In user mode,                             [16]
                            01FF   490                                         ; suppress echo of termination char         [16]
                            01FF   491                                         ; because VMS sometimes echoes it and       [16]
                            01FF   492                                         ; sometimes doesn't.                        [16]
                            01FF   493          $QIOW_S , -                    ; Read the user's response                  [16]
                            01FF   494                  L^DS$GW_TTIN, - ; Channel number                                [16][19]
                            01FF   495                  #IO$_READVBLK!IO$M_TRMNOECHO, - ; Function                      [16]
                            01FF   496                  P1 = (R7), -           ; Input buffer address                   [16]
                            01FF   497                  P2 = #1                ; Input buffer size                      [16]
                     21 11  0220   498          BrB     389$                   ;                                        [16]
                            0222   499 388$:   $QIOW_S , -                    ; Read the user's response               [16]
                            0222   500                  L^DS$GW_TTIN, - ; Channel number                                [16][19]
                            0222   501                  #IO$_READVBLK!IO$M_TRMNOECHO, - ; Function                      [16]
                            0222   502                  P1 = (R7), -           ; Input buffer address                   [16]
                            0222   503                  P2 = #0                ; S/A buffer works differently           [16]
                  0D 6E 91  0243   504 389$:   CmpB    (SP),#^X0D             ; CR ?                                   [16]
                     0F 13  0246   505          BEql    390$                   ; Yes                                    [16]
                     8F 93  0248   506          TstB    (SP)+                  ; No, pop the input buffer in case ^C    [16]
          00000000'EF    16 024A   507          JSb     L^KB_Check             ; Did we get ^C ?                        [16]
                     7E 94  0250   508          ClrB    -(SP)                  ; If we come back from KB_Check, then    [16]
                  57 6E DE  0252   509          MovAL   (SP), R7               ; it was not a ^C, so try again          [16]
```

```
                     99   11  0255   510          BrB     387$                    ;                                          [16]
                     8E   95  0257   511 390$:    TstB    (SP)+                   ; Pop the i    buffer                      [16]
                         0259   512              Br_If_Not_User  393$            ;                                          [16]
                         0261   513              $QIOW_S ,-                      ; Do an extra CRLF after prompt            [16]
                         0261   514                      L^DS$GW_TTOUT, -        ; Channel number                           [16]
                         0261   515                      S^#IO$_WRITEVBLK, -     ; Function                                 [16]
                         0261   516                      P1 = Page_String, -     ; Output buffer address                    [16]
                         0261   517                      P2 = #2                 ; Output buffer size                       [16]
                         0284   518 393$:    $QIOW_S ,-                      ; Do an extra CRLF after prompt            [16]
                         0284   519                      L^DS$GW_TTOUT, -        ; Channel number                           [16]
                         0284   520                      S^#IO$_WRITEVBLK, -     ; Function                                 [16]
                         0284   521                      P1 = Page_String, -     ; Output buffer address                    [16]
                         0284   522                      P2 = #2                 ; Output buffer size                       [16]
                         02A7   523
00000000'EF      94  02A7   524 395$:    ClrB    DS$GB_QIOACT            ; Clear QIO active                         [16]
00000009'EF      D4  02AD   525          ClrL    Line_Count              ; Reset line counter                       [16]
                 7E  7C  02B3   526          ClrQ    -(SP)                   ; P5/P6 zero                               [16]
                 7E  7C  02B5   527          ClrQ    -(SP)                   ; P4/P3 zero                               [16]
               FE35  31  02B7   528          BrW     310$                    ; Do another page                          [16]
                     04  02BA   529          ret                             ; Return                                   [09]
                         02BB   530
```

```
                     532          .Subtitle       DSX$Print Routine - Load type code byte
02BB    533 ;++
02BB    534 ; Functional Description:
02BB    535 ;
02BB    536 ;       This routine is the control point for Supervisor type coding.
02BB    537 ;       it sets the typecode byte to the proper value, and calls the
02BB    538 ;       desired PRINT routine.  Any print which does not go through
02BB    539 ;       this routine, and which does not set DS$GB_TYPECODE explicitly
02BB    540 ;       will have the default (current) type code [normally 0].
02BB    541 ;
02BB    542 ; Input Parameters:
02BB    543 ;
02BB    544 ;       4(ap)           Type code value (negative of typecode indicates
02BB    545 ;                       that code is 'sticky', and will not be cleared.
02BB    546 ;       8(ap)           Which PRINTx is desired
02BB    547 ;       12(ap)          Address of ASCIC format string
02BB    548 ;       16(ap)          First FAO argument (as in DSX$PRINTx routines)
02BB    549 ;         .
02BB    550 ;         .
02BB    551 ;         .
02BB    552 ;
02BB    553 ; Output Parameters:
02BB    554 ;
02BB    555 ;       none
02BB    556 ;
02BB    557 ; Implicit Inputs:
02BB    558 ;
02BB    559 ;       none
02BB    560 ;
02BB    561 ; Implicit Outputs:
02BB    562 ;
02BB    563 ;       DS$GL_TYPECODE  set to input value.
02BB    564 ;
02BB    565 ; Side effects:
02BB    566 ;
02BB    567 ;       none
02BB    568 ;
02BB    569 ; Status returns:
02BB    570 ;
02BB    571 ;       none
02BB    572 ;--
```

```
                        0000  02BB    574 .Entry  dsx$print, ^M<>                         ;                                       [09]
                          00    DD  02BD    575         pushl   #0                          ; Allocate a zero longword              [09]
                    50    04 AC    D0  02BF    576         movl    4(ap), r0               ; Copy type code                        [09]
                          50    D5  02C3    577         tstl    r0                          ; Check typecode                        [09]
                          05    18  02C5    578         bgeq    10$                         ; Branch if not sticky                  [09]
                       50   50    CE  02C7    579         mnegl   r0, r0                  ; If sticky, then negate and            [09]
                          6E    D6  02CA    580         incl    (sp)                        ; ... mark as sticky                    [09]
                                    02CC    581
            00000000'EF   50    90  02CC    582 10$:    movb    r0, L^ds$gb_typecode        ; Store the thing                       [09]
                                    02D3    583         case    src=8(ap),type=b,limit=#0, -    ; Case on the dispatch code         [09]
                                    02D3    584          displist=<-                        ;                                       [09]
                                    02D3    585             20$, -                          ; PRINTI                                [09]
                                    02D3    586             30$, -                          ; PRINTF                                [09]
                                    02D3    587             40$, -                          ; PRINTB                                [09]
                                    02D3    588             50$>                            ; PRINTX                                [09]
                                    02E0    589
                    50    50'AF    9E  02E0    590 20$:    movab   B^dsx$printi, r0            ; Address for PRINTI                    [09]
                          10    11  02E4    591         brb     60$                         ;                                       [09]
                                    02E6    592
                    50    38'AF    9E  02E6    593 30$:    movab   B^dsx$printf, r0            ; Address for PRINTF                    [09]
                          0A    11  02EA    594         brb     60$                         ;                                       [09]
                                    02EC    595
                    50    23'AF    9E  02EC    596 40$:    movab   B^dsx$printb, r0            ; Address for PRINTB                    [09]
                          04    11  02F0    597         brb     60$                         ;                                       [09]
                                    02F2    598
                    50    16'AF    9E  02F2    599 50$:    movab   B^dsx$printx, r0            ; Address for PRINTX                    [09]
                                    02F6    600
                08 AC    6C    02    C3  02F6    601 60$:    subl3   #2, (ap), 8(ap)             ; Shorten arg list                      [09]
                      60    08 AC    FA  02FB    602         callg   8(ap), (r0)             ; Call the routine                      [09]
                            06 8E    E8  02FF    603         blbs    (sp)+, 70$              ; Return if sticky type                 [09]
                         00000000'EF    94  0302    604         clrb    L^ds$gb_typecode        ; Otherwise clear it                    [09]
                                    0308    605
                                    04  0308    606 70$:    ret                             ; Return                                [09]
```

```
0309    608          .Subtitle       DSX$PrintX Routines - Extended print routines
0309    609 ;++
0309    610 ; Functional Description:
0309    611 ;
0309    612 ;       DSX$PRINTS      ROUTINE TO PRINT SUMMARY MESSAGES
0309    613 ;       DSX$PRINTX      ROUTINE TO PRINT EXTENDED MESSAGES
0309    614 ;       DSX$PRINTB      ROUTINE TO PRINT BASIC MESSAGES
0309    615 ;       DSX$PRINTF      ROUTINE TO PRINT FORCED MESSAGES
0309    616 ;       DSX$PRINTI      ROUTINE TO PRINT EVEN IN APT MODE
0309    617 ;
0309    618 ; Calling Sequence:
0309    619 ;
0309    620 ;       Procedure Call Thru Vectors In Entry
0309    621 ;
0309    622 ; Input Parameters:
0309    623 ;
0309    624 ;       (AP)            = NUMBER OF PARAMETERS + 1
0309    625 ;       4(AP)           = ADDRESS OF COUNTED ASCII FORMAT STRING
0309    626 ;       8(AP)           = FIRST PARAMETER
0309    627 ;       (AP)[(AP)]      = LAST PARAMETER
0309    628 ;
0309    629 ; Implicit Inputs:
0309    630 ;
0309    631 ;       NONE
0309    632 ;
0309    633 ; Output Parameters:
0309    634 ;
0309    635 ;       NONE
0309    636 ;
0309    637 ; Implicit Outputs:
0309    638 ;
0309    639 ;       NONE
0309    640 ;
0309    641 ; Completion Codes:
0309    642 ;
0309    643 ;       NONE
0309    644 ;
0309    645 ; Side Effects:
0309    646 ;
0309    647 ;       NONE
0309    648 ;
0309    649 ;--
```

```
                        0309   651          .ENABL  LSB
                        0309   652
                0000    0309   653 .ENTRY  DSX$PRINTS,^M<>                          ; ENTRY MASK
                        030B   654
                        030B   655          Br_If_Not_IES   30$                     ; If summary not inhibited, go  [11]
                        0313   656 ;        BBC     #DSA$V_IES,L^DSA$GL_FLAGS,30$   ; If summary not inhibited go
        0131     31     0313   657          BRW     PRINT_X                         ; SUMMARY INHIBITED
                        0316   658
                0000    0316   659 .ENTRY  DSX$PRINTX,^M<>
                        0318   660
                        0318   661          Br_If_Not_IE3   10$                     ; Inhibit Level-3 flag          [11]
                        0320   662 ;        BBC     #DSA$V_IE3,L^DSA$GL_FLAGS,10$   ; INHIBIT LEVEL-3 FLAG
        0124     31     0320   663          BRW     PRINT_X                         ; LEVEL-3 MESSAGES INHIBITED
                        0323   664
                0000    0323   665 .ENTRY  DSX$PRINTB,^M<>
                        0325   666
                        0325   667 10$:     Br_If_Not_IE2   20$                     ; Inhibit Level-2 flag          [11]
                        032D   668 ;10$:     BBC     #DSA$V_IE2,L^DSA$GL_FLAGS,20$   ; INHIBIT LEVEL-2 FLAG
        0117     31     032D   669          BRW     PRINT_X                         ; LEVEL-2 INHIBITED
                        0330   670
                        0330   671 20$:     Br_If_Not_IE1   30$                     ; Inhibit Level-1 flag          [11]
                        0338   672 ;20$:     BBC     #DSA$V_IE1,L^DSA$GL_FLAGS,30$   ; INHIBIT LEVEL-1 FLAG
        010C     31     0338   673          BRW     PRINT_X                         ; LEVEL-1 INHIBITED...
                        033B   674                                                  ; ... => ALL INHIBITED
                        033B   675
                0000    033B   676 .ENTRY  DSX$PRINTF,^M<>
                        033D   677
                        033D   678 30$:     Br_If_Not_Apt   40$                     ; If not in APT mode, proceed   [11]
                        0345   679          Br_If_Not_NoRpt 40$                     ; If not supressing output, go  [11]
                        034D   680 ;30$:     BBC     #DSA$V_APT, -                   ; IF NOT IN APT MODE PROCEED
                        034D   681 ;                 L^DSA$GL_FLAGS,40$
                        034D   682 ;         BBC     #DSA$V_NORPT, -                 ; APT MODE, IF NOT SUPPRESSING ...
                        034D   683 ;                 L^DSA$GL_FLAGS,40$              ; ... OUTPUT, branch
        00F7     31     034D   684          BRW     PRINT_X                         ; YES, IN APT MODE AND OUTPUT ...
                        0350   685                                                  ; ... SUPPRESSED.
                        0350   686
                0000    0350   687 .ENTRY  DSX$PRINTI,^M<>
                        0352   688
    50  0000FE00'EF     DE     0352   689 40$:     MOVAL   L^DSA$GL_FLAGS,R0               ; POINT TO FLAGS LONGWORD
      5E    0001'CE     9E     0359   690          MOVAB   -DS$K_BUFSIZ+1(SP),SP           ; Allocate space                [09]
00000004'EF    01 AE     9E     035E   691          MOVAB   1(SP), L^QBUFF+4                ;                               [09]
7E  60   F7FFFFFF 8F     CB     0366   692          BICL3   #^CDSA$M_NORPT,(R0),-(SP)       ; SAVE STATE OF DSA$M_NORPT
            60   6E     CA     036E   693          BICL2   (SP),(R0)                       ; CLEAR NORPT FLAG
                        0371   694          Set_Output                              ; Set OUTPUT flag               [11]
                        0379   695
    7E  04 AC    01     C1     0379   696          ADDL3   #1,4(AP),-(SP)          ; COPY STRING ORIGIN
      7E   04 BC     9A     037E   697          MOVZBL  @4(AP),-(SP)            ; COPY CONTROL STRING LENGTH
           08 AC     DF     0382   698          PUSHAL  8(AP)                   ; ADDRESS OF INSERT LIST
     00000000'EF     7F     0385   699          PUSHAQ  L^QBUFF                 ; ADDRESS OF OUTPUT BUFFER DESCRIPTOR   [07]
     00000000'EF     3F     038B   700          PUSHAW  L^DS$GL_BUFLEN          ; ADDRESS TO RETURN STRING LENGTH       [07]
           0C AE     7F     0391   701          PUSHAQ  12(SP)                  ; ADDRESS OF CONTROL STRING DESCRIPTOR
00000000'9F    06     FB     0394   702          CALLS   #6,@#SYS$FAOL           ; DO THE EDITING
                        039B   703                                                                                     [18]
        01   50     D1     039B   704          CMPL    R0,#SS$_NORMAL          ; ARE PRINT MACRO'S OK, CORRECT PARAM.  [18]
             03     13     039E   705          BEQL    45$                     ;                                       [18]
          00A4   31     03A0   706          BRW     PRINT_X                 ; EXIT IF NOT CORRECT.                  [18]
                        03A3   707 45$:                                     ;                                       [18]
```

```
                                03A3   708          Br_If_Not_Binary        60$      ; Skip funny stuff if not binary          [10]
          00000000'EF   D6      03AB   709          incl    L^ds$gl_buflen           ; Increment length of final string        [09]
          00000004'EF   D7      03B1   710          decl    L^qbuff+4                ; Decrment address                        [09]
     50   00000000'EF   9E      03B7   711          movab   L^ds$gb_typecode, r0     ; Get address of type code                [09]
00000004'FF   60   90      03BE   712          movb    (r0), @L^qbuff+4         ; Set first byte of buffer = type code    [09]
                                03C5   713
     50   0000FE68'EF   7E      03C5   714 60$:     MOVAQ   L^DSA$GQ_MSGPTR,R0       ; POINT TO TEST DESCRIPTOR
     50   00000000'EF   3C      03CC   715          MOVZWL  L^DS$GL_BUFLEN,(R0)+     ; STRING LENGTH TO MAILBOX                [07]
     51   00000004'EF   DE      03D3   716          moval   L^Qbuff+4, r1            ; Get address of Qbuff                    [09]
          60   61   D0      03DA   717          movl    (r1), (r0)               ; Buffer adr to mailbox                   [09]
                                03DD   718
                                03DD   719 ;
                                03DD   720 ; vEBUG ***
                                03DD   721 ;
     :F 00000085'EF   E9      03DD   722          blbc    L^Debug_the_sucker, 68$ ; Skip debug stuff if flag clear          [99]
                                03E4   723          Br_If_Not_Binary        68$      ; Only do debug if BINARY is set          [10]
               1F   BB      03EC   724          pushr   #^M<r0,r1,r2,r3,r4>      ; Make debug transparent                  [99]
          54   6E   9E      03EE   725          movab   (sp), r4                 ; Save current sp                         [99]
               12   11      03F1   726          brb     62$                      ; Skip format string                      [99]
                                03F3   727
5A 21 3D 70 74 5B 000003FB'010E0000'  03F3   728 61$:     .ascid  "[tp=!ZB]!/"             ; Format string (with <CR><LF>)           [10]
                  2F 21 5D 42   0401
                                0405   729
          5E   0A   C2      0405   730 62$:     subl2   #10, sp                  ; Allocate space on stack                 [99]
               6E   9F      0408   731          pushab  (sp)                     ; Set it's address                        [99]
          7E   0A   9A      040A   732          movzbl  #10, -(sp)               ; And it's length                         [99]
          53   6E   9E      040D   733          movab   (sp), r3                 ; And remember it                         [99]
     7E   00   B1      9A      0410   734          movzbl  @(r1), -(sp)             ; Final argument for FAO                  [99]
          00   B1      94      0414   735          clrb    @(r1)                    ; Clear it out to avoid tty problems      [99]
               63   9F      0417   736          pushab  (r3)                     ; Where to put length                     [99]
               63   9F      0419   737          pushab  (r3)                     ; Address of result buffer                [99]
     D5   AF   9F      041B   738          pushab  61$                      ; Address of format string                [99]
00000000'GF   04   FB      041E   739          calls   #4, G^sys$fao            ; FAO it                                  [99]
          7E   63   7D      0425   740          movq    (r3), -(sp)              ; Push address of result string           [99]
     FC1A CF   02   FB      0428   741          calls   #2, dsx$type_out         ; Print it                                [99]
          5E   64   9E      042D   742          movab   (r4), sp                 ; Restore initial stack                   [99]
               1F   BA      0430   743          popr    #^M<r0,r1,r2,r3,r4>      ; Restore registers                       [99]
                                0432   744
                                0432   745 68$:                                  ;                                         [99]
                                0432   746
                                0432   747 ;
                                0432   748 ; DEBUG ***
                                0432   749 ;
               61   DD      0432   750          pushl   (r1)                     ; Push address of buffer                  [09]
:E   00000000'EF   3C      0434   751          movzwl  L^ds$gl_buflen, -(sp)    ; And length of same                      [09]
     FC07 CF   02   FB      043B   752          calls   #2, dsx$type_out         ; Do the actual output                    [09]
                                0440   753
0000FE00'EF   8E   C8      0440   754 90$:     BISL2   (SP)+,L^DSA$GL_FLAGS     ; RESTORE STATE OF SUPPRESS FLAG
                                0447   755
                                0447   756 PRINT_X:
     00000000'EF   16      0447   757          jsb     L^kb_check               ; Check for Control-C                     [08]
               04   044D   758          RET
                                044E   759
                                044E   760          .DSABL  LSB
                                044E   761
                                044E   762 ;+
                                044E   763 ; Cancel current input QIO operation and reset ^C handler
```

```
                     044E    764 ;-
                     044E    765
                     044E    766 RESET_INPUT:
                     044E    767         $CANCEL_S CHAN=L^DS$GW_TTIN     ; Cancel I/O on input channel          [07]
                     045C    768         $QIOW_S CHAN=L^DS$GW_TTIN, -    ; Reset ^C handler                     [07]
                     045C    769                 FUNC=#IO$_SETMODE!IO$M_CTRLCAST, -
                     045C    770                 P1=RCTRLC
                  05 0481    771         RSB                             ; Return
```

```
                        0482    773          .Subtitle       DSX$CvtReg Routine - Mnemonic register conversion routine
                        0482    774 ;++
                        0482    775 ; FUNCTIONAL DESCRIPTION:
                        0482    776 ;
                        0482    777 ;       ASSOCIATES REGISTER BITS AND/OR FIELDS WITH ASCII MNEMONICS
                        0482    778 ;       GIVEN AN ASCII CONTROL STRING AND THE REGISTER CONTENTS
                        0482    779 ;       Instead of the mnemonic address a code may specified directing
                        0482    780 ;       CVTREG to use a predefined mnemonic string.  This allows
                        0482    781 ;       transportable diagnostics to format generic register values,
                        0482    782 ;       i.e. UNIBUS adapter MAP registers.
                        0482    783 ;
                        0482    784 ; CALLING SEQUENCE:
                        0482    785 ;
                        0482    786 ;       CALLG   ARGLIST,@#DS$CVTREG
                        0482    787 ;       CALLS   #11,@#DS$CVTREG
                        0482    788 ;
                        0482    789 ; INPUT PARAMETERS:
                        0482    790 ;
                        0482    791 ;        4(AP)  = STARTING BIT POSITION (MSB)
                        0482    792 ;        8(AP)  = REGISTER CONTENTS
                        0482    793 ;       12(AP)  = COUNTED ASCII BIT MNEMONIC STRING ADDRESS OR CODE
                        0482    794 ;       16(AP)  = OUTPUT BUFFER ADDRESS
                        0482    795 ;       20(AP)  = OUTPUT BUFFER SIZE
                        0482    796 ;       24(AP)  = OPTIONAL ARGUMENTS FOR MNEMONIC STRING DIRECTIVES (UP TO 6)
                        0482    797 ;
                        0482    798 ; IMPLICIT INPUTS:
                        0482    799 ;
                        0482    800 ;       NONE
                        0482    801 ;
                        0482    802 ; OUTPUT PARAMETERS:
                        0482    803 ;
                        0482    804 ;       NONE
                        0482    805 ;
                        0482    806 ; IMPLICIT OUTPUTS:
                        0482    807 ;
                        0482    808 ;       NONE
                        0482    809 ;
                        0482    810 ; COMPLETION CODES:
                        0482    811 ;
                        0482    812 ;       DS$_NORMAL
                        0482    813 ;       DS$_PROGERR
                        0482    814 ;
                        0482    815 ; SIDE EFFECTS:
                        0482    816 ;
                        0482    817 ;       R1 = LENGTH OF OUTPUT STRING + COUNT (NOT TRUNCATED)
                        0482    818 ;
                        0482    819 ; REGISTER USAGE:
                        0482    820 ;
                        0482    821 ;        R0  CURRENT BIT POSITION
                        0482    822 ;        R1  CURRENT OUTPUT BUFFER LENGTH
                        0482    823 ;        R2  CURRENT OUTPUT BUFFER POINTER
                        0482    824 ;        R3  CURRENT MNEMONIC STRING POINTER
                        0482    825 ;        R4  MNEMONIC STRING TERMINATOR (LAST BYTE ADDRESS + 1)
                        0482    826 ;        R5  FIELD SIZE
                        0482    827 ;        R6  EMUL OVERFLOW, EDIV QUOTIENT AND GENERAL USAGE
                        0482    828 ;        R7  EDIV DIVIDEND (HIGH ORDER)
                        0482    829 ;        R8  RADIX FOR EDIV
```

```
0482   830 ;        R9  LOOP CONTROL
0482   831 ;       R10  LOOP CONTROL
0482   832 ;       R11  OPTIONAL ARGUMENT POINTER
0482   833 ;--
```

```
                        0FFC  0482   835 .ENTRY DSX$CVTREG,^M<R2,R3,R4,R5,R6,R7,R8 ,R10,R11>
                              0484   836
                 6C     0B    D1  0484   837         CMPL    #11,(AP)                ; CK ARGLIST VALIDITY
                        03    13  0487   838         BEQL    10$                     ; OK, PROCEED
                     0199     31  0489   839         BRW     330$                    ; SOMETHING WRONG !
                              048C   840
              20     04 AC    D1  048C   841 10$:     CMPL    4(AP),#32               ; CHECK MSB POSITION
                        03    1F  049C   842         BLSSU   30$                     ; OK, PROCEED
                              0492   843
                     0190     31  0492   844 20$:     BRW     330$                    ;
                              0495   845                                            ; SOMETHING WRONG !
              50     04 AC    D0  0495   846 30$:     MOVL    4(AP),R0                ; GET STARTING BIT POSITION
                 51     01    D0  0499   847         MOVL    #1,R1                   ; SET UP OUTPUT LENGTH (COUNT BYTE)
           52    01     10 AC    C1  049C   848         ADDL3   16(AP),#1,R2            ; GET OUTPUT BUFFER ADDRESS
              5B     18 AC    DE  04A1   849         MOVAL   24(AP),R11              ; Get address of optional args
              53     0C AC    D0  04A5   850         MOVL    12(AP),R3               ; GET MNEMONIC STRING ADDRESS
                        1B    18  04A9   851         BGEQ    40$                     ; Branch if string address
           53    FFFFFFFC 8F    D1  04AB   852         CMPL    #CVT$_MAX,R3            ; Last assigned code
                        DE    14  04B2   853         BGTR    20$                     ; Unassigned code in R3
                 53     53    CE  04B4   854         MNEGL   R3,R3                   ; Negate it making it positive
        53    FFFFFFFC'EF43    D0  04B7   855         MOVL    CVT$A_CODELIST-4[R3],R3 ; Address of list, use one origin
              5B     04 A3    DE  04BF   856         MOVAL   4(R3),R11               ; Address of AUX arglist
                 53     63    D0  04C3   857         MOVL    (R3),R3                 ; Point to ASCIC mnemonic string
                              04C6   858
                 54     83    9A  04C6   859 40$:     MOVZBL  (R3)+,R4                ; GET MNEMONIC STRING LENGTH
                 54     53    C0  04C9   860         ADDL2   R3,R4                   ; GET MNEMONIC STRING TERMINATOR
                              04CC   861                                            ;
                    06     BB  04CC   862 50$:     PUSHR   #^M<R1,R2>              ; CURRENT OUTPUT LENGTH & ADR
              01    51     D1  04CE   863         CMPL    R1,#1                   ; LOOK AT OUTPUT LENGTH
                    0F     13  04D1   864         BEQL    70$                     ; SKIP IF EMPTY
           14 AC    51     D1  04D3   865         CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                    04     19  04D7   866         BLSS    60$                     ; NO, PROCEED
           52    10 AC    D0  04D9   867         MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                              04DD   868
              82    2C     90  04DD   869 60$:     MOVB    #^A',',(R2)+            ; STICK IN A SEPARATOR
                    51     D6  04E0   870         INCL    R1                      ; BUMP OUTPUT LENGTH
                              04E2   871                                            ;
              54    53     D1  04E2   872 70$:     CMPL    R3,R4                   ; MNEMONIC STRING EXHAUSTED ?
                    46     1E  04E5   873         BGEQU   120$                    ; YES, GO CHECK LAST BIT
              2C    63     91  04E7   874         CMPB    (R3),#^A','             ; END OF THIS MNEMONIC ?
                    41     13  04EA   875         BEQL    120$                    ; YES, GO CHECK THE BIT
                              04EC   876                                            ;
           14 AC    51     D1  04EC   877         CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                    04     19  04F0   878         BLSS    80$                     ; NO, PROCEED
           52    10 AC    D0  04F2   879         MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                              04F6   880                                            ;
              82    63     90  04F6   881 80$:     MOVB    (R3),(R2)+              ; COPY CHAR TO OUTPUT BUFFER
                    51     D6  04F9   882         INCL    R1                      ; BUMP OUTPUT LENGTH
              3D    83     91  04FB   883         CMPB    (R3)+,#^A'='            ; FIELD DIRECTIVE ?
                    E2     12  04FE   884         BNEQ    70$                     ; NO, LOOP FOR NEXT CHARACTER
                    55     D4  0500   885         CLRL    R5                      ; YES, INITIALIZE FIELD SIZE
                              0502   886                                            ;
              54    53     D1  0502   887 90$:     CMPL    R3,R4                   ; MNEMONIC STRING EXHAUSTED ?
                    03     1F  0505   888         BLSSU   100$                    ; NO, PROCEED
                 011B     31  0507   889         BRW     330$                    ; YES, BAD PLACE TO RUN OUT OF GAS
                              050A   890                                            ;
           5E 8F    63     91  050A   891 100$:    CMPB    (R3),#^A'^'             ; NUMERIC DIRECTIVE ?
```

```
                        2A   13  050E   892          BEQL    140$                    ; YES, GO EXTRACT A FIELD
                  40 8F 63   91  0510   893          CMPB    (R3),#^A'@'             ; INDIRECT DIRECTIVE ?
                        24   13  0514   894          BEQL    140$                    ; YES, GO EXTRACT A FIELD
                             0516   895                                              ;
                     56 83   9A  0516   896          MOVZBL  (R3)+,R6                ; GET NEXT CHARACTER
                     56 30   C2  0519   897          SUBL2   #^A'0',R6               ; CONVERT TO BINARY NUMERIC
                        0C   19  051C   898          BLSS    110$                    ; NO GOOD, WENT NEGATIVE
                     0A 56   D1  051E   899          CMPL    R6,#10                  ; DECIMAL DIGIT ?
                        07   18  0521   900          BGEQ    110$                    ; NO, BAD CHARACTER
               55    56 0A   55  7A  0523   901          EMUL    R5,#10,R6,R5            ; EXTEND AND ADD IN DIGIT
                        D8   11  0528   902          BRB     90$                     ; GO BACK FOR NEXT DIGIT
                             052A   903
                   00F8      31  052A   904 110$:    BRW     330$                    ; SOMETHING WRONG !
                             052D   905
               05 08 AC    50   E0  052D   906 120$:    BBS     R0,8(AP),130$           ; ASSOCIATED BIT SET ?
                        06   BA  0532   907          POPR    #^M<R1,R2>              ; NO, REMOVE CURRENT MNEMONIC
                   00B6      31  0534   908          BRW     290$                    ; AND GO CLEAN UP
                             0537   909
                   00B0      31  0537   910 130$:    BRW     280$                    ; ELSE PRESERVE OUTPUT AS IS
                             053A   911
                        50   D6  053A   912 140$:    INCL    R0                      ; ADJUST FOR POSITIONING
                     50 55   C2  053C   913          SUBL2   R5,R0                   ; POSITION TO RIGHTMOST BIT
               56    08 AC 55   50  EF  053F   914          EXTZV   R0,R5,8(AP),R6          ; GET THE FIELD FROM THE REGISTER
                  40 8F    63   91  0545   915          CMPB    (R3),#^A'@'             ; WAS IT AN INDIRECT DIRECTIVE ?
                        21   12  0549   916          BNEQ    170$                    ; NO, PROCEED WITH DIRECT CONVERSION
                        53   D6  054B   917          INCL    R3                      ; SKIP OVER DIRECTIVE DELIMITER
                     57 8B   D0  054D   918          MOVL    (R11)+,R7               ; ADDRESS OF STRING LIST
                  57    04 A746  D0  0550   919          MOVL    4(R7)[R6],R7            ; ADDRESS OF SELECTED STRING
                     59 87   9A  0555   920          MOVZBL  (R7)+,R9                ; LENGTH OF SELECTED STRING
                             0558   921                                              ;
                  14 AC 51   D1  0558   922 150$:    CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                        04   19  055C   923          BLSS    160$                    ; NO, PROCEED
                  52    10 AC D0  055E   924          MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                             0562   925                                              ;
                     82 87   90  0562   926 160$:    MOVB    (R7)+,(R2)+             ; COPY CHAR TO OUTPUT BUFFER
                        51   D6  0565   927          INCL    R1                      ; BUMP OUTPUT LENGTH
                        EE 59 F5  0567   928          SOBGTR  R9,150$                 ; LOOP TILL DONE
                        74   11  056A   929          BRB     270$                    ; THEN GO CLEAN UP
                             056C   930                                              ;
                        53   D6  056C   931 170$:    INCL    R3                      ; SKIP OVER DIRECTIVE DELIMITER
                  58 8F 63   91  056E   932          CMPB    (R3),#^A'X'             ; HEX CONVERSION ?
                        0C   12  0572   933          BNEQ    180$                    ; NO, TRY DECIMAL
                     58 10   D0  0574   934          MOVL    #16,R8                  ; SET UP RADIX
               59    55 01   C3  0577   935          SUBL3   #1,R5,R9                ; ADJUST FIELD SIZE
                     59 04   C6  057B   936          DIVL2   #4,R9                   ; CONVERT TO HEX DIGIT COUNT -1
                        27   11  057E   937          BRB     210$                    ; AND SKIP
                             0580   938
                  44 8F 63   91  0580   939 180$:    CMPB    (R3),#^A'D'             ; DECIMAL CONVERSION ?
                        0C   12  0584   940          BNEQ    190$                    ; NO, TRY OCTAL
                     58 0A   D0  0586   941          MOVL    #10,R8                  ; SET UP RADIX
               59    03 55   C5  0589   942          MULL3   R5,#3,R9                ; ADJUST FIELD SIZE
                     59 0A   C6  058D   943          DIVL2   #10,R9                  ; CONVERT TO DECIMAL DIGIT COUNT -1
                        15   11  0590   944          BRB     210$                    ; AND SKIP
                             0592   945                                              ;
                  4F 8F 63   91  0592   946 190$:    CMPB    (R3),#^A'O'             ; OCTAL CONVERSION ?
                        0C   12  0596   947          BNEQ    200$                    ; NO, GIVE UP
                     58 08   D0  0598   948          MOVL    #8,R8                   ; SET UP RADIX
```

```
                59   55   01   C3  059B   949          SUBL3   #1,R5,R9                ; ADJUST FIELD SIZE
                     59   03   C6  059F   950          DIVL2   #3,R9                   ; CONVERT TO OCTAL DIGIT COUNT -1
                          03   11  05A2   951          BRB     210$                    ; AND SKIP
                                   05A4   952
                        007E   31  05A4   953 200$:     BRW     330$                    ; SOMETHING WRONG !
                                   05A7   954                                          ;
                     7E   29   90  05A7   955 210$:     MOVB    #^A')',-(SP)            ; FOR CLARIFICATION,
                     7E   83   90  05AA   956          MOVB    (R3)+,-(SP)             ; RADIX DIRECTIVE IS
                     7E   28   90  05AD   957          MOVB    #^A'(',-(SP)            ; APPENDED TO NUMERIC DISPLAY
                5A   03   59   C1  05B0   958          ADDL3   R9,#3,R10               ; COPY ADJUSTED LOOP COUNT
                                   05B4   959                                          ;
                          57   D4  05B4   960 220$:     CLRL    R7                      ; DON'T NEED HIGH ORDER
             57   56   56   58   7B  05B6   961          EDIV    R8,R6,R6,R7             ; NEXT DIGIT TO R7
                     0A   57   D1  05BB   962          CMPL    R7,#10                  ; ALPHA DIGIT (I.E., HEX) ?
                          05   19  05BE   963          BLSS    230$                    ; NO, PROCEED
                     57   37   C0  05C0   964          ADDL2   #^A'A'-10,R7            ; YES, CONVERT TO 'HEX' ASCII
                          03   11  05C3   965          BRB     240$                    ; AND SKIP
                                   05C5   966                                          ;
                     57   30   C0  05C5   967 230$:     ADDL2   #^A'0',R7               ; CONVERT TO ASCII DIGIT
                                   05C8   968                                          ;
                7E   57   90  05C8   969 240$:     MOVB    R7,-(SP)                ; STACK ASCII DIGIT
                  E6 59   F4  05CB   970          SOBGEQ  R9,220$                 ; LOOP TILL DONE
                                   05CE   971                                          ;
             14 AC   51   D1  05CE   972 250$:     CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                     04   19  05D2   973          BLSS    260$                    ; NO, PROCEED
             52   10 AC   D0  05D4   974          MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                                   05D8   975                                          ;
                82   8E   90  05D8   976 260$:     MOVB    (SP)+,(R2)+             ; STORE THE NEXT CHARACTER
                     51   D6  05DB   977          INCL    R1                      ; BUMP OUTPUT LENGTH
                  EE 5A   F4  05DD   978          SOBGEQ  R10,250$                ; LOOP TILL DONE
                                   05E0   979                                          ;
             54   53   D1  05E0   980 270$:     CMPL    R3,R4                   ; MNEMONICS EXHAUSTED ?
                  16   1E  05E3   981          BGEQU   300$                    ; YES, JUST SKIP OUT QUIETLY
             2C   63   91  05E5   982          CMPB    (R3),#^A','             ; LOOKING AT A COMMA ?
                  3B   12  05E8   983          BNEQ    330$                    ; NO, THEN SOMETHING'S WRONG
                                   05EA   984                                          ;
                5E   08   C0  05EA   985 280$:     ADDL2   #8,SP                   ; PRESERVE OUTPUT AS IS
                                   05ED   986                                          ;
                     53   D6  05ED   987 290$:     INCL    R3                      ; SKIP COMMA IN MNEMONICS
                     50   D7  05EF   988          DECL    R0                      ; SHIFT TO NEXT BIT POSITION
                     08   19  05F1   989          BLSS    300$                    ; DONE, GO FINISH UP
                54   53   D1  05F3   990          CMPL    R3,R4                   ; PERHAPS MNEMONICS EXHAUSTED
                     03   1E  05F6   991          BGEQU   300$                    ; IT IS, JUST MEANS DONE
                   FED1   31  05F8   992          BRW     50$                     ; ELSE GO BACK TO DO SOME MORE
                                   05FB   993                                          ;
  00000100 8F      51   D1  05FB   994 300$:     CMPL    R1,#1@8                 ; COUNTED FORMAT OVERFLOW ?
                     08   14  0602   995          BGTR    310$                    ; YES, GO FIX IT
             14 AC   51   D1  0604   996          CMPL    R1,20(AP)               ; CALLER'S BUFFER OVERFLOW ?
                     02   14  0608   997          BGTR    310$                    ; YES, GO FIX IT
                     25   11  060A   998          BRB     350$                    ; BOTH OK, DO NORMAL THING
                                   060C   999                                          ;
  00000100 8F   14 AC   D1  060C  1000 310$:     CMPL    20(AP),#1@8             ; FIND THE SMALLER ONE
                     07   19  0614  1001          BLSS    320$                    ; CALLER'S BUFFER IS .LSS. 256
          10 BC   FF 8F   90  0616  1002          MOVB    #<1@8>-1,@16(AP)        ; STORE MAX COUNT, BUFFER IS BIG ENOUGH
                     0B   11  061B  1003          BRB     340$                    ; AND SKIP OUT
                                   061D  1004                                          ;
  10 BC   14 AC   01   83  061D  1005 320$:     SUBB3   #1,20(AP),@16(AP)       ; COPY CALLER'S MAX LENGTH
```

```
                      03   11  0623  1006          BRB     340$                    ; AND SKIP OUT
                                 0625  1007                                        ;
                   10 BC   94  0625  1008 330$:     CLRB    @16(AP)                 ; INDICATE NO OUTPUT
                                 0628  1009                                        ;
           50  00660020 8F   D0  0628  1010 340$:     MOVL    #DS$_PROGERR,R0         ; INDICATE A PROBLEM
                      0C   11  062F  1011          BRB     RCVTREGX                ; AND SKIP OUT
                                 0631  1012                                        ;
            10 BC   51   01   83  0631  1013 350$:     SUBB3   #1,R1,@16(AP)           ; STORE STRING LENGTH
           50  00660001 8F   D0  0636  1014          MOVL    #DS$_NORMAL,R0          ; INDICATE SUCCESS
                                 063D  1015                                        ;
                                 063D  1016 RCVTREGX:                              ;
                      04   063D  1017          RET                               ; RETURN TO CALLER
                                 063E  1018
                                 063E  1019          .END
```

```
$$.TMP1                         =  00000001     D
$$.TMP2                         =  00000060
$$N                             =  00000002     D
$$T1                            =  00000001     D
BIT...                          =  00000003     D
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_SIZE                         00000444     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_LAST                         00000024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEST                         00000020     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       0000003C     D
CLI$V_ADAPTER                   =  00000018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BREAK                     =  0000000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
CR                              =  0000000D     D
CRD$K_CRD_INITIALIZATION        =  00000000   G D
CRD$K_DS_CONTROL_C              =  00000001   G D
CRD$K_DS_FLAGS                  =  00000000   G D
CRD$K_FAILURE                   =  00000000     D
CRD$K_HOOK_POINT                =  00000000   G D
CRD$K_LAST_ACCESS_CODE          =  00000002   G D
CRD$K_LAST_DS_VARIABLE          =  00000002   G D
CRD$K_LAST_FUNCTION             =  00000002   G D
CRD$K_LAST_HOOK_POINT           =  00000001   G D
CRD$K_READ                      =  00000000   G D
CRD$K_SUCCESS                   =  00000001     D
CRD$K_TYPECODE                  =  00000001   G D
CRD$K_WRITE                     =  00000001   G D
CVT$A_CODELIST                     00000000 R   D   03
CVT$T_MBAMAP                       ********     X   03
CVT$T_MBASR                        ********     X   03
CVT$T_UBADPR                       ********     X   03
CVT$T_UBAMAP                       ********     X   03
CVT$_MAX                        =  FFFFFFFC       D
CVT$_MBAMAP                     =  FFFFFFFC       D
CVT$_MBASR                      =  FFFFFFFD       D
CVT$_UBADPR                     =  FFFFFFFF       D
CVT$_UBAMAP                     =  FFFFFFFE       D
DEBUG_THE_SUCKER                   00000085 RG  D   02
DS$GA_CRD_DS_INTERFACE             ********     X   04
DS$GB_QIOACT                       ********     X   04
DS$GB_TYPECODE                     ********     X   04
DS$GB_VDS_DEBUG                    00000085 RG  D   02
DS$GB_WIDTH                        ********     X   04
DS$GL_BUFLEN                       ********     X   04
DS$GL_FLAGS                        ********     X   04
DS$GL_PAGE                         0000000D RG  D   02
DS$GT_BUFFER                       ********     X   02
DS$GW_TTIN                         ********     X   04
DS$GW_TTOUT                        ********     X   04
DS$K_BUFSIZ                        ********     X   02
DS$K_ERROR                      =  00000002       D
DS$K_NORMAL                     =  00000001       D
DS$K_PRINTB                     =  00000002       D
DS$K_PRINTF                     =  00000001       D
DS$K_PRINTI                     =  00000000       D
DS$K_PRINTX                     =  00000003       D
DS$K_SEVERE                     =  00000004       D
DS$K_SUBSYS                     =  00000066       D
DS$K_TYPE_ABORT_PROGRAM         =  00000014       D
DS$K_TYPE_ABORT_TEST            =  00000013       D
DS$K_TYPE_COMMAND_ERR           =  00000015       D
DS$K_TYPE_COMMAND_OUT           =  0000001C       D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A       D
DS$K_TYPE_DS_PROMPT             =  00000001       D
DS$K_TYPE_DS_START              =  0000001D       D
DS$K_TYPE_ERRDEV                =  00000008       D
DS$K_TYPE_ERRHARD               =  00000006       D
DS$K_TYPE_ERROR_BODY            =  00000009       D
DS$K_TYPE_ERROR_END             =  0000000A       D
DS$K_TYPE_ERRPREP               =  0000001B       D
DS$K_TYPE_ERRSOFT               =  00000007       D
DS$K_TYPE_ERRSUP                =  00000004       D
DS$K_TYPE_ERRSYS                =  00000005       D
```

```
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$RAB_OUTPUT                   = ********     X    04
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 00000008     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INSFMEM                     = 00660050     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
```

```
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SIZ                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$M_NORPT                     = 08000000     D
DSA$V_APT                       = 0000001F     D
DSA$V_BINARY                    = 00000001     D
DSA$V_CRD_AUTOTEST_OFF          = 0000000B     D
DSA$V_CRD_MENUTEST_OFF          = 00000010     D
DSA$V_IE1                       = 00000004     D
DSA$V_IE2                       = 00000005     D
DSA$V_IE3                       = 00000006     D
DSA$V_IES                       = 00000007     D
DSA$V_NORPT                     = 0000001B     D
DSA$V_USER                      = 0000001C     D
DSV$SETPAGE                       00000000 RG  D  04
DSV$SHOWPAGE                      00000027 RG  D  04
DSX$CVTREG                        00000482 RG  D  04
DSX$PRINT                         000002BB RG  D  04
DSX$PRINTB                        00000323 RG  D  04
DSX$PRINTF                        0000033B RG  D  04
DSX$PRINTI                        00000350 RG  D  04
DSX$PRINTS                        00000309 RG  D  04
DSX$PRINTX                        00000316 RG  D  04
DSX$TYPE_OUT                      00000047 RG  D  04
FALSE                           = 00000000     D
IO$M_CTRLCAST                     ********     X  04
IO$M_TRMNOECHO                    ********     X  04
IO$_READVBLK                      ********     X  04
IO$_SETMODE                       ********     X  04
IO$_WRITEVBLK                     ********     X  04
KB_CHECK                          ********     X  04
LF                              = 0000000A     D
LINE_COUNT                        00000009 RG  D  02
OFF                             = 00000000     D
ON                              = 00000001     D
PAGE_STRING                       00000058 R   D  02
PRINT_X                           00000447 R   D  04
QBUFF                             00000000 R   D  02
RAB$L_RBF                       = 00000028     D
RAB$W_RSZ                       = 00000022     D
RCTRLC                            ********     X  04
RCVTREGX                          0000063D R   D  04
RESET_INPUT                       0000044E R   D  04
SIZ...                          = 00000001     D
SS$_NORMAL                      = 00000001     D
SYS$CANCEL                        ********    GX  04
SYS$FAO                           ********     X  04
SYS$FAOL                          ********     X  04
SYS$PUT                           ********    GX  04
SYS$QIOW                          ********    GX  04
TRUE                            = 00000001     D
T_PAGE                            00000011 R   D  02
T_SHOW_PAGE                       00000034 R   D  02
WIDTH_TOO_LARGE                   00000008 R   D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000086 (134.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000010 (16.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	0000063E (1598.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$.TMP1	=00000001	381 (1)	381 (1)
$$.TMP2	=00000060	381 (1)	381 (1)
$$N	=00000002	290 (1)	#-245 (1) 290 (1)
$$T1	=00000001	770 (1)	477 (1) 484 (1) 497 (1) 503 (1) 517 (1) 522 (1) 770 (1)
BIT...	=00000003	144 (1)	138 (1) 140 (1) 141 (1) 143 (1) 144 (1)
CLI$K_BUFSIZ	=00000100	145 (1)	145 (1)
CLI$K_SIZE	00000444	145 (1)	
CLI$L_ADDRESS	00000018	145 (1)	
CLI$L_COMMAND	00000004	145 (1)	
CLI$L_DATA	0000001C	145 (1)	#-234 (1)
CLI$L_FLAGS	00000000	145 (1)	
CLI$L_LAST	00000024	145 (1)	
CLI$L_NEXT	00000030	145 (1)	
CLI$L_PASS	0000002C	145 (1)	
CLI$L_SUBT	00000028	145 (1)	
CLI$L_TEST	00000020	145 (1)	
CLI$Q_BUFQWD	00000034	145 (1)	
CLI$Q_FILE	00000008	145 (1)	
CLI$Q_SECTION	00000010	145 (1)	
CLI$Q_TIME	0000043C	145 (1)	
CLI$T_BUFFER	0000003C	145 (1)	
CLI$V_ADAPTER	=00000018	145 (1)	
CLI$V_ADR	=0000000B	145 (1)	
CLI$V_ASCII	=00000013	145 (1)	
CLI$V_BREAK	=0000000A	145 (1)	
CLI$V_BRIEF	=0000001B	145 (1)	
CLI$V_BYTE	=0000000D	145 (1)	
CLI$V_CLEAR	=00000002	145 (1)	
CLI$V_DEC	=00000010	145 (1)	
CLI$V_DEFAULT	=0000000C	145 (1)	
CLI$V_DEPOSIT	=00000019	145 (1)	
CLI$V_EVENT	=00000008	145 (1)	
CLI$V_EXAM	=00000005	145 (1)	
CLI$V_FLAGS	=00000009	145 (1)	
CLI$V_HEX	=00000012	145 (1)	
CLI$V_KERNEL	=00000017	145 (1)	
CLI$V_LOAD	=00000006	145 (1)	
CLI$V_LONG	=0000000F	145 (1)	
CLI$V_NOTNUF	=00000001	145 (1)	
CLI$V_OCT	=00000011	145 (1)	
CLI$V_PREG	=0000001A	145 (1)	
CLI$V_QA	=00000007	145 (1)	
CLI$V_QACKLOOPLOOPS	=0000001C	145 (1)	
CLI$V_QAERRORPRINTS	=0000001B	145 (1)	
CLI$V_QAMULTIPLEPASS	=0000001F	145 (1)	
CLI$V_QASUBTESTLOOPS	=0000001E	145 (1)	
CLI$V_QATESTLOOPS	=0000001D	145 (1)	
CLI$V_REG	=00000014	145 (1)	

```
CLI$V_REQUIRED                  =00000000     145     (1)
CLI$V_RUN                       =00000015     145     (1)
CLI$V_SET                       =00000003     145     (1)
CLI$V_SHOW                      =00000004     145     (1)
CLI$V_VALSEC                    =00000016     145     (1)
CLI$V_WORD                      =0000000E     145     (1)
CR                              =0000000D     166     (1)       168     (1)
CRD$K_CRD_INITIALIZATION        =00000000     144     (1)
CRD$K_DS_CONTROL_C              =00000001     144     (1)
CRD$K_DS_FLAGS                  =00000000     144     (1)
CRD$K_FAILURE                   =00000000     144     (1)
CRD$K_HOOK_POINT                =00000000     144     (1)
CRD$K_LAST_ACCESS_CODE          =00000002     144     (1)
CRD$K_LAST_DS_VARIABLE          =00000002     144     (1)
CRD$K_LAST_FUNCTION             =00000002     144     (1)
CRD$K_LAST_HOOK_POINT           =00000001     144     (1)
CRD$K_READ                      =00000000     144     (1)
CRD$K_SUCCESS                   =00000001     144     (1)
CRD$K_TYPECODE                  =00000001     144     (1)     #-353     (1)
CRD$K_WRITE                     =00000001     144     (1)
CVT$A_CODELIST                   00000000-R   187     (1)     #-855     (1)
CVT$T_MBAMAP                     00000000-XR                    191     (1)
CVT$T_MBASR                      00000000-XR                    190     (1)
CVT$T_UBADPR                     00000000-XR                    188     (1)
CVT$T_UBAMAP                     00000000-XR                    189     (1)
CVT$_MAX                        =FFFFFFFC     140     (1)     #-852     (1)
CVT$_MBAMAP                     =FFFFFFFC     140     (1)       140     (1)
CVT$_MBASR                      =FFFFFFFD     140     (1)
CVT$_UBADPR                     =FFFFFFFF     140     (1)
CVT$_UBAMAP                     =FFFFFFFE     140     (1)
DEBUG_THE_SUCKER                 00000085-R   172     (1)     #-722     (1)
DS$GA_CRD_DS_INTERFACE           00000000-XR                    354     (1)
DS$GB_QIOACT                     00000000-XR                  #-389     (1)     #-488     (1)     #-524     (1)
DS$GB_TYPECODE                   00000000-XR                  #-342     (1)     #-352     (1)     #-582     (1)     #-604     (1)
                                                                711     (1)
DS$GB_VDS_DEBUG                  00000085-R   171     (1)
DS$GB_WIDTH                      00000000-XR                  #-416     (1)
DS$GL_BUFLEN                     00000000-XR                    700     (1)     #-709     (1)     #-715     (1)     #-751     (1)
DS$GL_FLAGS                      00000000-XR                    359     (1)       694     (1)
DS$GL_PAGE                       0000000D-R   161     (1)     #-237     (1)     #-285     (1)     #-409     (1)     #-412     (1)
                                                              #-449     (1)
DS$GT_BUFFER                     00000000-XR                    155     (1)
DS$GW_TTIN                       00000000-XR                  #-497     (1)     #-503     (1)     #-767     (1)     #-770     (1)
DS$GW_TTOUT                      00000000-XR                  #-463     (1)     #-477     (1)     #-484     (1)     #-517     (1)
                                                              #-522     (1)
DS$K_BUFSIZ                      00000000-XR                    154     (1)       690     (1)
DS$K_ERROR                      =00000002     138     (1)
DS$K_NORMAL                     =00000001     138     (1)
DS$K_PRINTB                     =00000002     143     (1)
DS$K_PRINTF                     =00000001     143     (1)     #-245     (1)     #-290     (1)
DS$K_PRINTI                     =00000000     143     (1)
DS$K_PRINTX                     =00000003     143     (1)
DS$K_SEVERE                     =00000004     138     (1)
DS$K_SUBSYS                     =00000066     138     (1)       138     (1)
DS$K_TYPE_ABORT_PROGRAM         =0000C014     143     (1)
DS$K_TYPE_ABORT_TEST            =00000013     143     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     143     (1)     #-245     (1)
```

```
DS$K_TYPE_COMMAND_OUT           =00000016      143     (1)     #-290     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A      143     (1)     #-341     (1)
DS$K_TYPE_DS_PROMPT             =00000001      143     (1)
DS$K_TYPE_DS_START              =0000001D      143     (1)
DS$K_TYPE_ERRDEV                =00000008      143     (1)
DS$K_TYPE_ERRHARD               =00000006      143     (1)
DS$K_TYPE_ERROR_BODY            =00000009      143     (1)
DS$K_TYPE_ERROR_END             =0000000A      143     (1)
DS$K_TYPE_ERRPREP               =0000001B      143     (1)
DS$K_TYPE_ERRSOFT               =00000007      143     (1)
DS$K_TYPE_ERRSUP                =00000004      143     (1)
DS$K_TYPE_ERRSYS                =00000005      143     (1)
DS$K_TYPE_ERR_HALT              =0000000D      143     (1)
DS$K_TYPE_EXCEPTION             =0000000C      143     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      143     (1)
DS$K_TYPE_FIRST_PASS            =00000011      143     (1)
DS$K_TYPE_GENERAL               =00000000      143     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      143     (1)
DS$K_TYPE_NO_TESTS              =00000012      143     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      143     (1)
DS$K_TYPE_PROGRAM_END           =00000010      143     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      143     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      143     (1)
DS$K_TYPE_QIO_INVADP            =00000024      143     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      143     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      143     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      143     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      143     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      143     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      143     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      143     (1)
DS$K_TYPE_START_ERR             =00000018      143     (1)
DS$K_TYPE_START_LIST            =00000025      143     (1)
DS$K_TYPE_SUMMARY               =0000000E      143     (1)
DS$K_TYPE_USER_PROMPT           =00000002      143     (1)
DS$K_WARNING                    =00000000      138     (1)
DS$M_ABRTFLG                    =00000040      141     (1)
DS$M_BADTIME                    =00100000      141     (1)
DS$M_BATCH                      =00400000      141     (1)
DS$M_BRKCLR                     =00001000      141     (1)
DS$M_BRKPT                      =00000800      141     (1)
DS$M_CHARFLG                    =00000100      141     (1)
DS$M_CMDFLG                     =00000080      141     (1)
DS$M_CTRLC                      =00000001      141     (1)
DS$M_CTRLO                      =00010000      141     (1)
DS$M_DEVFLG                     =00000200      141     (1)
DS$M_DISABLCC                   =01000000      141     (1)
DS$M_DONFLG                     =00002000      141     (1)
DS$M_ERRFLG                     =00000010      141     (1)
DS$M_EXCEPT                     =00080000      141     (1)
DS$M_EXETST                     =00040000      141     (1)
DS$M_HLTFLG                     =00000008      141     (1)
DS$M_LODFLG                     =00000002      141     (1)
DS$M_MEMMGT                     =00008000      141     (1)
DS$M_OUTPUT                     =00800000      141     (1)
DS$M_RUBFLG                     =00000020      141     (1)
DS$M_SCRIPT                     =00200000      141     (1)
```

```
DS$M_SETIMR                     =02000000      141     (1)
DS$M_STRFLG                     =00000004      141     (1)
DS$M_SUBT                       =00004000      141     (1)
DS$M_SYSFLG                     =00000400      141     (1)
DS$M_TIMRON                     =00020000      141     (1)
DS$RAB_OUTPUT                    00000000-XR                    368     (1)
DS$V_ABRTFLG                    =00000006      141     (1)
DS$V_BADTIME                    =00000014      141     (1)
DS$V_BATCH                      =00000016      141     (1)     #-359   (1)
DS$V_BRKCLR                     =0000000C      141     (1)
DS$V_BRKPT                      =0000000B      141     (1)
DS$V_CHARFLG                    =00000008      141     (1)
DS$V_CMDFLG                     =00000007      141     (1)
DS$V_CTRLC                      =00000000      141     (1)
DS$V_CTRLO                      =00000010      141     (1)
DS$V_DEVFLG                     =00000009      141     (1)
DS$V_DISABLCC                   =00000018      141     (1)
DS$V_DONFLG                     =0000000D      141     (1)
DS$V_ERRFLG                     =00000004      141     (1)
DS$V_EXCEPT                     =00000013      141     (1)
DS$V_EXETST                     =00000012      141     (1)
DS$V_HLTFLG                     =00000003      141     (1)
DS$V_LODFLG                     =00000001      141     (1)
DS$V_MEMMGT                     =0000000F      141     (1)
DS$V_OUTPUT                     =00000017      141     (1)     #-694   (1)
DS$V_RUBFLG                     =00000005      141     (1)
DS$V_SCRIPT                     =00000015      141     (1)
DS$V_SETIMR                     =00000019      141     (1)
DS$V_STRFLG                     =00000002      141     (1)
DS$V_SUBT                       =0000000E      141     (1)
DS$V_SYSFLG                     =0000000A      141     (1)
DS$V_TIMRON                     =00000011      141     (1)
DS$_ARITH                       =006600D0      138     (1)
DS$_ASBE                        =00660118      138     (1)
DS$_BADLINK                     =006600F0      138     (1)
DS$_BADTYPE                     =006600E8      138     (1)
DS$_BIIC                        =00660120      138     (1)
DS$_CHME                        =006600A8      138     (1)
DS$_CHMK                        =006600E0      138     (1)
DS$_DEVNAME                     =00660108      138     (1)
DS$_ERROR                       =00660002      138     (1)
DS$_FHWE                        =00660068      138     (1)
DS$_FRAGBUF                     =00660080      138     (1)
DS$_ICBUSY                      =006600C8      138     (1)
DS$_ICERR                       =006600C0      138     (1)
DS$_IHWE                        =00660060      138     (1)
DS$_ILLCHAR                     =00660018      138     (1)
DS$_ILLPAGCNT                   =00660078      138     (1)
DS$_ILLUNIT                     =00660100      138     (1)
DS$_INSFMEM                     =00660050      138     (1)
DS$_IPL2HI                      =006600B8      138     (1)
DS$_IVADDR                      =00660040      138     (1)
DS$_IVVECT                      =00660038      138     (1)
DS$_KRNLSTK                     =00660090      138     (1)
DS$_LOGIC                       =00660070      138     (1)
DS$_MCHK                        =00660088      138     (1)
DS$_MMOFF                       =00660058      138     (1)
```

```
DS$_NEEDUNIT                    =006600F8      138     (1)
DS$_NODE                        =00660128      138     (1)
DS$_NOPCS                       =00660110      138     (1)
DS$_NORMAL                      =00660001      138     (1)     #-1014    (1)
DS$_NOSUPPORT                   =006600B1      138     (1)
DS$_NOTDON                      =00660030      138     (1)
DS$_NOTIMP                      =006600B0      138     (1)       138     (1)
DS$_NULLSTR                     =00660010      138     (1)
DS$_OVERFLOW                    =00660008      138     (1)
DS$_POWER                       =00660098      138     (1)
DS$_PROGERR                     =00660020      138     (1)     #-1010    (1)
DS$_SEVERE                      =00660004      138     (1)
DS$_TRANSL                      =006600A0      138     (1)
DS$_TRUNCATE                    =00660028      138     (1)
DS$_UNEXPINT                    =006600D8      138     (1)
DS$_VASFULL                     =00660048      138     (1)
DS$_WARNING                     =00660000      138     (1)       138     (1)
DSA$GL_FLAGS                     0000FE00                         346     (1)       347     (1)       372     (1)       373     (1)
                                                                  398     (1)       399     (1)       437     (1)       487     (1)
                                                                  512     (1)       655     (1)       661     (1)       667     (1)
                                                                  671     (1)       678     (1)       679     (1)       689     (1)
                                                                  708     (1)       723     (1)     #-754     (1)
DSA$GQ_MSGPTR                    0000FE68                         714     (1)
DSA$M_NORPT                     =08000000                       #-692     (1)
DSA$V_APT                       =0000001F                       #-678     (1)
DSA$V_BINARY                    =00000001                       #-708     (1)     #-723     (1)
DSA$V_CRD_AUTOTEST_OFF          =0000000B                       #-346     (1)     #-372     (1)     #-398     (1)
DSA$V_CRD_MENUTEST_OFF          =00000010                       #-347     (1)     #-373     (1)     #-399     (1)
DSA$V_IE1                       =00000004                       #-671     (1)
DSA$V_IE2                       =00000005                       #-667     (1)
DSA$V_IE3                       =00000006                       #-661     (1)
DSA$V_IE5                       =00000007                       #-655     (1)
DSA$V_NORPT                     =0000001B                       #-673     (1)
DSA$V_USER                      =0000001C                       #-437     (1)     #-487     (1)     #-512     (1)
DSV$SETPAGE                      00000000-R     233     (1)
DSV$SHOWPAGE                     00000027-R     284     (1)
DSX$CVTREG                       00000482-R     835     (1)
DSX$PRINT                        000002BB-R     574     (1)       245     (1)       290     (1)
DSX$PRINTB                       00000323-R     665     (1)       596     (1)
DSX$PRINTF                       0000033B-R     676     (1)       593     (1)
DSX$PRINTI                       00000350-R     687     (1)       590     (1)
DSX$PRINTS                       00000309-R     653     (1)
DSX$PRINTX                       00000316-R     659     (1)       599     (1)
DSX$TYPE_OUT                     00000047-R     339     (1)       741     (1)       752     (1)
FALSE                           =00000000       144     (1)
IO$M_CTRLCAST                    00000000-XR                     #-770     (1)
IO$M_TRMNOECHO                   00000000-XR                     #-497     (1)     #-503     (1)
IO$_READVBLK                     00000000-XR                     #-497     (1)     #-503     (1)
IO$_SETMODE                      00000000-XR                     #-770     (1)
IO$_WRITEVBLK                    00000000-XR                     #-462     (1)     #-477     (1)     #-484     (1)     #-517     (1)
                                                                 #-522     (1)
KB_CHECK                         00000000-XR                       507     (1)       757     (1)
LF                              =0000000A       167     (1)       168     (1)
LINE_COUNT                       00000009-R     159     (1)     #-412     (1)     #-449     (1)     #-525     (1)
OFF                             =00000000       144     (1)
ON                              =00000001       144     (1)
PAGE_STRING                      00000058-R     168     (1)       477     (1)       484     (1)       517     (1)       522     (1)
```

```
PRINT_X                         00000447-R   756   (1)   #-657   (1)   #-663   (1)   #-669   (1)   #-673   (1)
                                                          #-684   (1)   #-706   (1)
QBUFF                           00000000-R   154   (1)   #-691   (1)     699   (1)   #-710   (1)   #-712   (1)
                                                            716   (1)
RAB$L_RBF                      =00000028                  #-380   (1)
RAB$W_RSZ                      =00000022                  #-379   (1)
RCTRLC                          00000000-XR               #-770   (1)
RCVTREGX                        0000063D-R  1016   (1)   #-1011  (1)
RESET_INPUT                     0000044E-R   766   (1)   #-390   (1)
SIZ...                         =00000001    141   (1)     141   (1)
SS$_NORMAL                     =00000001                  #-704   (1)
SYS$CANCEL                      00000000-XR               #-767   (1)
SYS$FAO                         00000000-XR               #-739   (1)
SYS$FAOL                        00000000-XR               #-702   (1)
SYS$PUT                         00000000-XR               #-381   (1)
SYS$QIOW                        00000000-XR               #-465   (1)     477   (1)     484   (1)     497   (1)
                                                          #-503   (1)     517   (1)     522   (1)     770   (1)
TRUE                           =00000001    144   (1)
T_PAGE                          00000011-R   162   (1)     245   (1)
T_SHOW_PAGE                     00000034-R   164   (1)     290   (1)
WIDTH_TOO_LARGE                 00000008-R   157   (1)   #-417   (1)   #-443   (1)   #-471   (1)   #-478   (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANCEL_S	1	767 (1)	767 (1)
$CRD_LITERALS	2	144 (1)	144 (1)
$D1_PRINT_S	1	245 (1)	245 (1) 290 (1)
$DEF	1	144 (1)	
$DEFINI	1	137 (1)	137 (1) 139 (1) 142 (1)
$DS_CVTDEF	1	140 (1)	140 (1)
$DS_DSADEF	5	139 (1)	139 (1)
$DS_DSDEF	2	138 (1)	138 (1)
$DS_TYPEDEF	4	143 (1)	143 (1)
$EQU	1	144 (1)	138 (1) 140 (1) 143 (1) 144 (1)
$EQULS1	1	144 (1)	138 (1) 140 (1) 143 (1) 144 (1)
$EQULST	1	138 (1)	138 (1) 140 (1) 143 (1) 144 (1)
$GBLINI	2		138 (1) 140 (1) 141 (1) 143 (1) 144 (1)
$PRINT	2	243 (1)	243 (1) 287 (1)
$PUSHADR	1	477 (1)	477 (1) 484 (1) 497 (1) 503 (1) 517 (1) 522 (1) 770 (1)
$PUSHTWO	1	477 (1)	477 (1) 484 (1) 497 (1) 503 (1) 517 (1) 522 (1) 770 (1)
$PUT	1	381 (1)	381 (1)
$QIOPUSH	1	477 (1)	477 (1) 484 (1) 497 (1) 503 (1) 517 (1) 522 (1) 770 (1)
$QIOW_S	1	473 (1)	473 (1) 480 (1) 493 (1) 499 (1) 513 (1) 518 (1) 768 (1)
$RABDEF	6	142 (1)	142 (1)
$RMSCALL	2	381 (1)	381 (1)
$SSDEF	21	137 (1)	137 (1)
$VIELD	1		141 (1)
$VIELD1	1	144 (1)	141 (1)
BR_IF_CRD_AUTOTEST_OFF	1	346 (1)	346 (1) 372 (1) 398 (1)
BR_IF_CRD_MENUTEST_OFF	1	347 (1)	347 (1) 373 (1) 399 (1)
BR_IF_NOT_APT	1	678 (1)	678 (1)
BR_IF_NOT_BATCH	1	359 (1)	359 (1)
BR_IF_NOT_BINARY	1	708 (1)	708 (1) 723 (1)
BR_IF_NOT_IE1	1	671 (1)	671 (1)
BR_IF_NOT_IE2	1	667 (1)	667 (1)
BR_IF_NOT_IE3	1	661 (1)	661 (1)
BR_IF_NOT_IES	1	655 (1)	655 (1)
BR_IF_NOT_NORPT	1	679 (1)	679 (1)
BR_IF_NOT_USER	1	437 (1)	437 (1) 487 (1) 512 (1)
CASE	1	583 (1)	583 (1)
CLIDEF	3	145 (1)	145 (1)
DSFDEF	3	141 (1)	141 (1)
SET_OUTPUT	1	694 (1)	694 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.12	00:00:00.28
Command processing	137	00:00:00.84	00:00:02.35
Pass 1	1029	00:00:16.19	00:00:23.07
Symbol table sort	21	00:00:01.43	00:00:01.58
Pass 2	355	00:00:03.56	00:00:08.25
Symbol table output	44	00:00:00.26	00:00:00.27
Psect synopsis output	8	00:00:00.03	00:00:00.04
Cross-reference output	154	00:00:01.27	00:00:01.84
Assembler run totals	1786	00:00:23.71	00:00:37.69

The working set limit was 1000 pages.
90011 bytes (176 pages) of virtual memory were used buffer the intermediate code.
There were 50 pages of symbol table space allocated hold 863 non-local and 84 local symbols.
1019 source lines were read in Pass 1, producing 0 object records in Pass 2.
112 pages of virtual memory were used to define 39 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	5
DRB1:[DS.WORK]DS.MLB;218	15
DRB1:[DS.WORK]CRD.MLB;12	1
SYS$SYSROOT:[SYSLIB]LIB_MLB;1	1
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	36

1116 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) PRINT/UPDA=(PRINT.UPD,PRINT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

```
;  0001  0     %title '*** PROBE Check address accessibility'
;  0002  0     module probe (                  ! Routine to test accessibility of addresses
;  0003  0                   ident = '01-05'
;  0004  0                   ) =
;  0005  0     !
;  0006  0     ! Copyright (c) 1979, 1981, 1983
;  0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;  0008  0     !
;  0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;  0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;  0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;  0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;  0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;  0014  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;  0015  0     ! REMAIN IN DEC.
;  0016  0     !
;  0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;  0018  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;  0019  0     ! CORPORATION.
;  0020  0     !
;  0021  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;  0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;  0023  0     !
;  0024  0     !++
;  0025  0     ! FACILITY: DIAGNOSTIC SUPERVISOR
;  0026  0     !
;  0027  0     ! ABSTRACT:
;  0028  0     !
;  0029  0     !       This routine tests the accessibility of an address.
;  0030  0     !
;  0031  0     ! AUTHOR:       Roger Riggs, CREATION DATE: 10-November-1979
;  0032  0     ! MODIFIED BY:
;  0033  0     !
;  0034  0     !                       - Dave Butenhof, 02-Jun-1981, version 6.4
;  0035  0     !       01              Alter library spec. for flexibility (logical name)
;  0036  0     !
;  0037  0     !       02              - Dave Butenhof, 23-Nov-1981, verison 6.5
;  0038  0     !                       Fix truncation error.
;  0039  0     !
;  0040  0     !       03              Bob Bergazzi    May 17, 1983    Version 6.11
;  0041  0     !                       Changed the order of searching libraries.
;  0042  0     !
;  0043  0     !       04              Bob Bergazzi    May 24, 1983    Version 6.12
;  0044  0     !                       Removed the call to DSR$FAULT_CLEAR from the SELECT statement
;  0045  0     !                       in the ACC$HANDLER routine. This routine was being called
;  0046  0     !                       twice, once at EXE_MCHK in EXCEPT and then here if the condition
;  0047  0     !                       was machine check.
;  0048  0     !
;  0049  0     !       05              Domenic Andella         7-Sep-83        Version 6.13
;  0050  0     !                       Deleted External reference to DSR$FAULT_CLEAR routine
;  0051  0     !--
;  0052  0
;  0053  1     begin
;  0054  1     !
;  0055  1     ! TABLE OF CONTENTS:
;  0056  1     !
;  0057  1
```

```
;    0058  1      forward routine
;    0059  1          dsx$probe,                          ! Routine to test address
;    0060  1          acc$handler;                        ! Exception handler routine for PROBE
;    0061  1
;    0062  1      !
;    0063  1      ! INCLUDE FILES:
;    0064  1      !
;    0065  1
;    0066  1      library '$DIAG';                !
;    0067  1
;    0068  1      library '$DS';                  !
;    0069  1
;    0070  1      library 'SYS$LIBRARY:STARLET.L32';
;    0071  1
;    0072  1      linkage
;    0073  1          quick = jsb;
;    0074  1
;    0075  1      !
;    0076  1      ! EXTERNAL REFERENCES:
;    0077  1      !
;    0078  1
;    0079  1      external
;    0080  1          io$gq_physical : vector [2, long]
;    0081  1              addressing_mode (long_relative);        ! Quad descriptor of I/O space
;    0082  1
;    0083  1      !
;    0084  1      ! PSECT DEFINITIONS:
;    0085  1      !
;    0086  1
;    0087  1      psect
;    0088  1          plit = data ( share, addressing_mode (long_relative));
;    0089  1
;    0090  1      psect
;    0091  1          code = sep ( read, write, share);
;    0092  1
```

```
;    0093  1     %sbttl 'DSX$PROBE'
;    0094  1
;    0095  1     global routine dsx$probe (address, length, unit) =
;    0096  1
;    0097  1     !++
;    0098  1     ! FUNCTIONAL DESCRIPTION:
;    0099  1     !
;    0100  1     !       This routine checks that the address specified can be
;    0101  1     !       accessed using instructions that access data with the specified
;    0102  1     !       length.  The Unit number reflects the device expected
;    0103  1     !       to respond to this address, (i.e. unibus adapter)
;    0104  1     !
;    0105  1     ! FORMAL PRAMETERS:
;    0106  1     !
;    0107  1     !       ADDRESS Address to be accessed
;    0108  1     !       LENGTH  Length of reference, (1,2,4)
;    0109  1     !       UNIT    Unit number of device responding to this address
;    0110  1     !
;    0111  1     ! IMPLICIT INPUTS:      NONE
;    0112  1     !
;    0113  1     ! IMPLICIT OUTPUTS:     NONE
;    0114  1     !
;    0115  1     ! COMPLETION CODES:
;    0116  1     !
;    0117  1     ! SIDE EFFECTS:
;    0118  1     !
;    0119  1     !--
;    0120  1
;    0121  2         begin
;    0122  2
;    0123  2         stacklocal
;    0124  2             ch_status : $ds_chs_decl;               ! Storage for channel status
;    0125  2
;    0126  2         local
;    0127  2             rc,
;    0128  2             scratch;
;    0129  2
;    0130  2         builtin
;    0131  2             fp;
;    0132  2
;    0133  2         .fp = acc$handler;                          ! Set exception handler
;    0134  2
;    0135  2         if .address<0, 30> gequ .io$gq_physical [0]
;    0136  2         then
;    0137  3             begin
;    0138  3
;    0139  3             if not (rc = $ds_channel (unit = .unit, func = chc$_clear)) then return .rc;
;    0140  3
;    0141  2             end;
;    0142  2
;    0143  2         case .length from 1 to 4 of
;    0144  2             set
;    0145  2
;    0146  2             [1] :
;    0147  2                 scratch = .(.address)<0, 8>;
;    0148  2
;    0149  2             [2] :
```

```
;    0150  2                    scratch = .(.address)<0, 16>;
;    0151  2
;    0152  2                [4] :
;    0153  2                    scratch = .(.address)<0, 32>;
;    0154  2
;    0155  2                [inrange, outrange] :
;    0156  2                    return ds$_error;
;    0157  2                tes;
;    0158  2
;    0159  2            if .address<0, 30> gequ .io$gq_physical [0]
;    0160  2            then
;    0161  3                begin
;    0162  3
;    0163  3                local
;    0164  3                    rc;
;    0165  3
;    0166  3                if not (rc = $ds_channel (unit = .unit, func = chc$_status, stsadr = ch_status)) then return .rc;
;    0167  3
;    0168  3                if .ch_status [chs$v_errany] neq 0 then return ds$_error;
;    0169  3
;    0170  2                end;
;    0171  2
;    0172  2            ss$_normal
;    0173  1            end;


                                                                .TITLE  PROBE *** PROBE Check address accessibility
                                                                .IDENT  \01-05\

                                                                .EXTRN  IO$GQ_PHYSICAL, DS$CHANNEL

                                                                .PSECT  SEP,  SHR,2

                                                000C 00000      .ENTRY  DSX$PROBE, Save R2,R3                           ; 0095
                                    53 00000000G  9F   9E 00002 MOVAB   @#DS$CHANNEL, R3                                ;
                                    52 00000000G  EF   9E 00009 MOVAB   IO$GQ_PHYSICAL, R2                              ;
                                    5E            08   C2 00010 SUBL2   #8, SP                                        ;
                                    6D      0000V CF   9E 00013 MOVAB   ACC$HANDLER, (FP)                               ; 0133
                62        04    AC  1E            00   ED 00018 CMPZV   #0, #30, ADDRESS, IO$GQ_PHYSICAL              ; 0135
                                                14   1F 0001E BLSSU   1$                                               ;
                                    00000000    9F   D4 00020 CLRL    @#^X00000000                                    ; 0139
                                                7E   7C 00026 CLRQ    -(SP)                                           ;
                                    7E          06   7D 00028 MOVQ    #6, -(SP)                                       ;
                                          0C    AC   DD 0002B PUSHL   UNIT                                            ;
                                    63          05   FB 0002E CALLS   #5, DS$CHANNEL                                  ;
                                    4E          50   E9 00031 BLBC    RC, 9$                                          ;
                      03            01    08    AC   CF 00034 1$:     CASEL   LENGTH, #1, #3                          ; 0143
0016                003E          0010          000A       00039 2$:  .WORD   3$-2$,-                                 ;
                                                                               4$-2$,-                                 ;
                                                                               7$-2$,-                                 ;
                                                                               5$-2$                                   ;
                                                34   11 00041       BRB     7$                                      ; 0156
                                    50    04    BC   9A 00043 3$:     MOVZBL  @ADDRESS, SCRATCH                       ; 0147
                                                0A   11 00047       BRB     6$                                      ;
                                    50    04    BC   3C 00049 4$:     MOVZWL  @ADDRESS, SCRATCH                       ; 0150
                                                04   11 0004D       BRB     6$                                      ;
                                    50    04    BC   D0 0004F 5$:     MOVL    @ADDRESS, SCRATCH                       ; 0153
```

```
          62       04    AC            1E              00  ED 00053 6$:     CMPZV   #0, #30, ADDRESS, IO$GQ_PHYSICAL          ; 0159
                                                       24  1F 00059         BLSSU   8$                                          ;
                                   00000000            9F  D4 0005B         CLRL    @#^X00000000                                ; 0166
                                                       7E  D4 00061         CLRL    -(SP)                                       ;
                                           04          AE  9F 00063         PUSHAB  CH_STATUS                                   ;
                                   7E                  07  7D 00066         MOVQ    #7, -(SP)                                   ;
                                           0C          AC  DD 00069         PUSHL   UNIT                                        ;
                                   63                  05  FB 0006C         CALLS   #5, DS$CHANNEL                              ;
                                   10                  50  E9 0006F         BLBC    RC, 9$                                      ;
                                   0F                  6E  93 00072         BITB    CH_STATUS, #15                              ; 0168
                                                       08  13 00075         BEQL    8$                                          ;
                                   50 00660002         8F  D0 00077 7$:     MOVL    #6684674, R0                                ;
                                                           04 0007E         RET                                                 ;
                                   50                  01  D0 0007F 8$:     MOVL    #1, R0                                      ; 0173
                                                           04 00082 9$:     RET                                                 ;

; Routine Size:  131 bytes,    Routine Base:  SEP + 0000

;   0174  1
```

```
;    0175  1     %sbttl 'ACC$HANDLER'
;    0176  1
;    0177  1     global routine acc$handler (signal, mechanism) =
;    0178  1
;    0179  1     !++
;    0180  1     ! FUNCTIONAL DESCRIPTION:
;    0181  1     !
;    0182  1     !       This condition handler routine can be used to field any exceptions
;    0183  1     !       that indicate memory is in accessable.
;    0184  1     !       It does an SYS$UNWIND to the level of the caller of the routine
;    0185  1     !       that generated the condition.  The conditions trapped are
;    0186  1     !       DS$_MCHK, SS$_ACCVIO, and SS$_TRANSL.  The value of the aborted
;    0187  1     !       routine will be one of these values.
;    0188  1     !
;    0189  1     ! FORMAL PARAMETERS:
;    0190  1     !
;    0191  1     !       SIGNAL          Address of standard signal array
;    0192  1     !       MECHANISM       Address of mechanism array
;    0193  1     !
;    0194  1     ! IMPLICIT INPUTS:      NONE
;    0195  1     !
;    0196  1     ! IMPLICIT OUTPUTS:     NONE
;    0197  1     !
;    0198  1     ! CONDITION CODES:
;    0199  1     !
;    0200  1     !       SS$_RESIGNAL    If condition not DS$_MCHK, DS$_TRANSL, SS$_ACCVIO
;    0201  1     !       Otherwise       The value is irrelevent.
;    0202  1     !
;    0203  1     !--
;    0204  1
;    0205  2         begin
;    0206  2
;    0207  2         map
;    0208  2             signal : ref block [, byte],
;    0209  2             mechanism : ref block [, byte];
;    0210  2
;    0211  2         select .signal [chf$l_sig_name] of
;    0212  2             set
;    0213  2
;    0214  2             [ds$_mchk, ss$_accvio, ds$_transl] :
;    0215  3                 begin
;    0216  3                 mechanism [chf$l_mch_savr0] = .signal [chf$l_sig_name];
;    0217  4                 $unwind ()
;    0218  2                 end;
;    0219  2
;    0220  2             [otherwise] :
;    0221  2                 ss$_resignal;
;    0222  2             tes
;    0223  2
;    0224  1         end;


                                                .EXTRN  SYS$UNWIND

                                0000C 00000     .ENTRY  ACC$HANDLER, Save R2,R3                 ; 0177
                     50    04   AC  D0 00002     MOVL    SIGNAL, R0                              ; 0211
```

```
                                50              04  C0 00006          ADDL2   #4, R0                                        ;
                                52              60  D0 00009          MOVL    (R0), R2                                    ;
                                53              01  D0 0000C          MOVL    #1, R3                                      ;
                                0C              52  D1 0000F          CMPL    R2, #12                                     ; 0214
                                                12  13 00012          BEQL    1$                                          ;
                    00660088    8F              52  D1 00014          CMPL    R2, #6684808                                ;
                                                09  13 00018          BEQL    1$                                          ;
                    006600A0    8F              52  D1 0001D          CMPL    R2, #6684832                                ;
                                                16  12 00024          BNEQ    2$                                          ;
                                                53  D4 00026 1$:      CLRL    R3                                          ;
                                50      08      AC  D0 00028          MOVL    MECHANISM, R0                               ; 0216
                             0C     A0          52  D0 0002C          MOVL    R2, 12(R0)                                  ;
                                                7E  7C 00030          CLRQ    -(SP)                                       ; 0217
                    00000000G   00              02  FB 00032          CALLS   #2, SYS$UNWIND                              ;
                                51              50  D0 00039          MOVL    R0, R1                                      ;
                                05              53  E9 0003C 2$:      BLBC    R3, 3$                                      ; 0220
                                50      0918    8F  3C 0003F          MOVZWL  #2328, R0                                   ;
                                                    04 00044 3$:      RET                                                 ; 0224

; Routine Size:  69 bytes,    Routine Base:  SEP + 0083

;   0225  1
;   0226  1      end
;   0227  1
;   0228  0      eludom


;                                   PSECT SUMMARY

;       Name                        Bytes                           Attributes

;  SEP                               200  NOVEC,  WRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                                 Library Statistics

;                                          --------  Symbols  --------      Pages       Processing
;       File                               Total     Loaded   Percent       Mapped      Time

;  DRB1:[DS.WORK]DIAG.L32;265               784         37         4          85        00:00.2
;  DRB1:[DS.WORK]DS.L32;159                 653          0         0          42        00:00.2
;  SYS$SYSROOT:[SYSLIB]STARLET.L32;60       9486         7         0         570        00:02.6


;                                   COMMAND QUALIFIERS
```

```
;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE PROBE

; Size:         200 code + 0 data bytes
; Run Time:        00:07.0
; Elapsed Time:    00:23.5
; Lines/CPU Min:    1965
; Lexemes/CPU-Min:  7120
; Memory Used:  61 pages
; Compilation Complete
```

Table of contents

(2) 45 DECLARATIONS
(3) 96 UDA50 Bootstrap device initialization
(4) 232 UDA50 Bootstrap driver QIO

```
                                0000     1          .TITLE  PUBTDRIVR - UDA50 BOOT DRIVER
                                0000    .1          .IDENT  '06-01'
-1                              0000     3
                                0000     4 ;
                                0000     5 ;*****************************************************************************
                                0000     6 ;*                                                                           *
                                0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
                                0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                                0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
                                0000    10 ;*                                                                           *
                                0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                                0000    12 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                                0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
                                0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                                0000    15 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
                                0000    16 ;*  TRANSFERRED.                                                             *
                                0000    17 ;*                                                                           *
                                0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
                                0000    19 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                                0000    20 ;*  CORPORATION.                                                             *
                                0000    21 ;*                                                                           *
                                0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                                0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                                0000    24 ;*                                                                           *
                                0000    25 ;*                                                                           *
                                0000    26 ;*****************************************************************************
                                0000    27 ;
                                0000    28
                                0000    29 ;++
                                0000    30 ; FACILITY:     BOOTS
                                0000    31 ;
                                0000    32 ; ABSTRACT:
                                0000    33 ;       This module contains the bootstrap device driver for the
                                0000    34 ;       UDA 50 disks.
                                0000    35 ;
                                0000    36 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
                                0000    37 ;
                                0000    38 ; AUTHOR:  Kerbey T. Altmann,   CREATION DATE:  20-Nov-1981
                                0000    39 ;
                                0000    40 ; MODIFIED BY:
                                0000    41 ;
                                0000    .1 ; [01] Dave Butenhof, 15-Apr-1982
                                0000    .2 ;       Modify to run under VDS. Partic, change $BOOT_DRIVER macro
                                0000    .3 ;       to exclude fields not supported by VDS, and have I/O entry
                                0000    .4 ;       call init routine @ each call (since diagnostics may be
                                0000    .5 ;       run, etc. between calls).
-1                              0000    43 ;--
```

```
                        45          .SBTTL  DECLARATIONS
                        46 ;
                        .1 ;        marcros
                        .2 ;
                        .3
                        .4          .MACRO  $PRDEF,$GBL
                        .5
                        .6          $DEFINI PR,$GBL
                        .7
                        .8
0000                    .9  $EQU    PR$_KSP 0
0000                   .10  $EQU    PR$_ESP 1
0000                   .11  $EQU    PR$_SSP 2
0000                   .12  $EQU    PR$_USP 3
0000                   .13  $EQU    PR$_ISP 4
0000                   .14  $EQU    PR$_P0BR        8
0000                   .15  $EQU    PR$_P0LR        9
0000                   .16  $EQU    PR$_P1BR        10
0000                   .17  $EQU    PR$_P1LR        11
0000                   .18  $EQU    PR$_SBR 12
0000                   .19  $EQU    PR$_SLR 13
0000                   .20  $EQU    PR$_PCBB        16
0000                   .21  $EQU    PR$_SCBB        17
0000                   .22  $EQU    PR$_IPL 18
0000                   .23  $EQU    PR$_ASTLVL      19
0000                   .24  $EQU    PR$_SIRR        20
0000                   .25  $EQU    PR$_SISR        21
0000                   .26  $EQU    PR$_ICCS        24
0000                   .27  $EQU    PR$_NICR        25
0000                   .28  $EQU    PR$_ICR 26
0000                   .29  $EQU    PR$_TODR        27
0000                   .30  $EQU    PR$_RXCS        32
0000                   .31  $EQU    PR$_RXDB        33
0000                   .32  $EQU    PR$_TXCS        34
0000                   .33  $EQU    PR$_TXDB        35
0000                   .34  $EQU    PR$_ACCS        40
0000                   .35  $EQU    PR$_ACCR        41
0000                   .36  $EQU    PR$_MAPEN       56
0000                   .37  $EQU    PR$_TBIA        57
0000                   .38  $EQU    PR$_TBIS        58
0000                   .39  $EQU    PR$_PME 61
0000                   .40  $EQU    PR$_SID 62
0000                   .41  $EQU    PR$_TBCHK       63
0000                   .42  $EQU    PR$V_SID_SN     0
0000                   .43  $EQU    PR$S_SID_SN     12
0000                   .44  $EQU    PR$V_SID_PL     12
0000                   .45  $EQU    PR$S_SID_PL     3
0000                   .46  $EQU    PR$V_SID_ECO    15
0000                   .47  $EQU    PR$S_SID_ECO    9
0000                   .48  $EQU    PR$V_SID_TYPE   24
0000                   .49  $EQU    PR$S_SID_TYPE   8
0000                   .50  $EQU    PR$_SID_TYP780  1
0000                   .51  $EQU    PR$_SID_TYP750  2
0000                   .52  $EQU    PR$_SID_TYP730  3
0000                   .53  $EQU    PR$_SID_TYP7VV  4
0000                   .54  $EQU    PR$_SID_TYPMAX  4
0000                   .55  $EQU    PR$_WCSA        44
```

```
                        0000    .56  $EQU    PR$_WCSD        45
                        0000    .57  $EQU    PR$_SBIFS       48
                        0000    .58  $EQU    PR$_SBIS        49
                        0000    .59  $EQU    PR$_SBISC       50
                        0000    .60  $EQU    PR$_SBIMT       51
                        0000    .61  $EQU    PR$_SBIER       52
                        0000    .62  $EQU    PR$_SBITA       53
                        0000    .63  $EQU    PR$_SBIQC       54
                        0000    .64  $EQU    PR$_CMIERR      23
                        0000    .65  $EQU    PR$_CSRS        28
                        0000    .66  $EQU    PR$_CSRD        29
                        0000    .67  $EQU    PR$_CSTS        30
                        0000    .68  $EQU    PR$_CSTD        31
                        0000    .69  $EQU    PR$_TBDR        36
                        0000    .70  $EQU    PR$_CADR        37
                        0000    .71  $EQU    PR$_MCESR       38
                        0000    .72  $EQU    PR$_CAER        39
                        0000    .73  $EQU    PR$_UBRESET     55
                        0000    .74  $EQU    PR$_PAMACC      64
                        0000    .75  $EQU    PR$_PAMLOC      65
                        0000    .76  $EQU    PR$_CSWP        66
                        0000    .77  $EQU    PR$_CRBT        67
                        0000    .78  $EQU    PR$_MCTL1       68
                        0000    .79  $EQU    PR$_MCTL2       69
                        0000    .80  $EQU    PR$_MGEN        70
                        0000    .81  $EQU    PR$_MTBER       71
                        0000    .82  $EQU    PR$_MEAR        72
                        0000    .83  $EQU    PR$_MDCR1       73
                        0000    .84  $EQU    PR$_MEDR        74
                        0000    .85  $EQU    PR$_MECCR       75
                        0000    .86
                        0000    .87          $DEFEND PR,$GBL,DEF
                        0000    .88
                        0000    .89          .ENDM   $PRDEF
                        0000    .90 ;
                        0000     47 ; INCLUDE FILES:
                        0000     48 ;
                        0000     49
                        0000     50          $BTDDEF                         ; Boot device types
                        0000     51          $IODEF                          ; I/O function codes
                        0000     52          $MSCPDEF                        ; MSCP definitions
                        0000     53          $PRDEF                          ; Processor registers
                        0000     54          $PTEDEF                         ; Page table entries
                        0000     55          $RPBDEF                         ; RPB offsets
                        0000     56          $SSDEF                          ; Status codes
                        0000     57          $UBADEF                         ; UBA definitions
                        0000     58          $UBIDEF                         ; 11/750 UBA definitions
                        0000     59          $VADEF                          ; Virtual addresses
                        0000     .1
-1                      0000     61
                        0000     62 ;
                        0000     63 ; EQUATED SYMBOLS:
                        0000     64 ;
                        0000     65
                00000000 0000    66          UDAIP   = 0
                00000002 0000    67          UDASA   = 2
                00000001 0000    68          GO      = 1
```

```
00008000  0000     69          OWN     = 1@15
0000000B  0000     70          S1      = 11
0000000E  0000     71          S4      = 14
000001EE  0000     72          UMR     = 494           ; Last-1 UNIBUS Mapping Register
          0000     73
          0000     74 ;
          0000     75 ; OWN STORAGE:
          0000     76 ;
          0000     77
          0000     78 ;
          0000     79 ; Boot driver table entry
          0000     80 ;
          0000     81
          00000000 .1 .Psect  Bootdrivr_2,    page    ; Define psect allocation
          0000     .2
          0000     82         $BOOT_DRIVER    DEVTYPE = BTD$K_UDA,-   ; Device type (UDA50)
          0000     83                         SIZE = UD_DRVSIZ,-      ; Driver size
          0000     84                         ADDR = START,-          ; Driver starting address
          0000     .1                         ENTRY = UD_DRIVER       ; Driver entry point
          0000     .2 ;                       UNIT_INIT = UD_INIT,-   ; Driver unit init entry
          0000     .3 ;                       DRIVRNAME = DSKDRVNAME,-; Driver disk name
          0000     .4 ;                       AUXDRNAME = PRTDRVNAME  ; Driver port name
          0000     .5
          0000     .6 START:
          0000     .7 ;DSKDRVNAME:
          0000     .8 ;        .ASCIC  /DUDRIVER.EXE/          ; Disk class driver filename
          0000     .9 ;PRTDRVNAME:
          0000    .10 ;        .ASCIC  /PUDRIVER.EXE/          ; Port driver filename
```

```
                                0000     96          .SBTTL  UDA50 Bootstrap device initialization
                                0000     97
                                0000     98 ;++
                                0000     99 ;
                                0000    100 ; Inputs:
                                0000    101 ;
                                0000    102 ;       R9 -->  RPB
-1                              0000    104 ;
                                0000    105 ; Outputs:
                                0000    106 ;
                                0000    107 ;       R0 - status code
                                0000    108 ;
                                0000    109 ;--
                                0000    110
                          01FC  C000     .1 .Entry UD_INIT, ^M<R2,R3,R4,R5,R6,R7,R8>
                                0002     .2
                                0002     .3         .ENABLE LSB
                                0002     .4
                     50   38   DB  0002     .5         MFPR    #PR$_MAPEN, R0          ; Get the mapping status
                                0005     .6 ;       BLBS    R0,10$                  ; If virtual, skip some set up
-6                              0005    117 ;
                                0005    118 ; Set up the SYSTEMID for the local UDA.
                                0005    119 ;
                                0005     .1 ;       MOVL    RPB$L_IOVEC(R9),R1      ; Point to iovec
                                0005     .2 ;       CLRB    B^<BOO$GB_UMR_DP-BOO$AL_VECTOR>(R1) ; Set for Direct Data Path
                                0005     .3 ;       INCL    VMB$L_FLAGS(AP)         ; Set a flag to load SCS code
                                0005     .4 ;       MOVL    RPB$L_BOOTR2(R9),-
                                0005     .5 ;                   VMB$B_SYSTEMID(AP)      ; Low 32 bits is CSR of UDA
                                0005     .6 ;       MOVW    RPB$L_BOOTR1(R9),-
                                0005      7 ;                   VMB$B_SYSTEMID+4(AP)    ; Hi 16 bits is TR of UBA
                                0005     .8 ;       BBSS    #47,VMB$B_SYSTEMID(AP),10$ ; Set bit 47
-8                              0005    128 ;
                                0005    129 ; Set up an interrupt vector.
                                0005    130 ;
                                0005     .1 ;10$:   MOVW    #<127*4>,RPB$W_R0UBVEC(R9) ; Use the highest possible
-1                              0005    132 ;
                                0005    133 ; Set up a UNIBUS mapping register(s) to cover the ring and buffers.
                                0005    134 ; To make things easy, we will grab the last two register (494 & 495).
                                0005    135 ; These registers are necessary since the ring area will be accessed by
                                0005    136 ; the controller which is a UNIBUS device that does not do any mapping.
                                0005    137 ;
                      0003DC00 8F    D0  0005    138         MOVL    #<UMR@9>, R6            ; Set up a constant
                               56        000B
                    52  0200'CF    9E  000C    139         MOVAB   W^INTTBL, R2            ; Get the address of the ring
                      52   77'AF    CB  0011    140         BICL3   B^BYTE_OFF, R2, R1      ; Get the byte offset in page
                                51        0015
                           51   56    C9  0016    141         BISL3   R6, R1, 2(R2)           ; Set in the UNIBUS addr
                              02 A2      0019
                        02 A2    10'   C0  001B    142         ADDL    S^#<RING-INTTBL>,2(R2)  ; Make it the ring address
                    52  15    09    EF  001F    143         EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2 ; Get the page frame
                                52        0023
                                          0024    144         ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
                    53   5C A940    D0  0024    145         MOVL    RPB$L_ADPPHY(R9)[R0],R3 ; Get correct pointer to UBA reg
                                          0029    146         ASSUME  RPB$L_CSRVIR EQ RPB$L_CSRPHY+4
                    57   54 A940    D0  0029    147         MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get correct address of device CSR
                           0C 50    E9  002E    148         BLBC    R0,20$                  ; If clr, then physical
                    52   50 B942    D0  0031    149         MOVL    @RPB$L_SVASFT(R9)[R2],R2; Virtual, get physical
```

```
                FFE00000 8F   CA  0036   150          BICL    #^C<PTE$M_PFN>,R2       ; Now a physical PFN
                      52      0003C
              54   0FB8 C3   DE  003D   151 20$:     MOVAL   UBA$L_MAP+<UMR*4>(R3),R4; Get the last two UMR's
                52   C0'AF   C9  0042   152          BISL3   B^VALID,R2,(R4)+        ; Set as valid w/PFN
                      84      0046
                      52      D6  0047   153          INCL    R2                      ; Set next page just in case
                52   C0'AF   C9  0049   154          BISL3   B^VALID,R2,(R4)         ; Set as valid w/PFN
                      64      004D
                              004E   155 ;
                              004E   156 ; Now go thru the ridiculously complicated startup sequence.  This is a
                              004E   157 ; fugue in four parts.
                              004E   158 ;
                   58   02    D0  004E   159          MOVL    #2,R8                   ; Make two tries at this
              53   0200'CF    9E  0051    .1 RETRY:  MOVAB   W^INTTBL, R3
-1                 50   0B    D0  C056   161          MOVL    #S1, R0                 ; Step flag
                      67      B4  0059   162          CLRW    UDAIP(R7)               ; Poke the controller's CSR
                              005B   163 ;
                              005B   164 ; Wait 100 microseconds.
                              005B   165 ;
                   52   1B    DB  005B   166 TIME:    MFPR    #PR$_TODR, R2           ; Current time
              52   03E8 C2    9E  005E   167          MOVAB   1000(R2), R2            ; Set for 10 seconds later
                54   02 A7    B0  0063   168 LOOP:    MOVW    UDASA(R7),R4            ; Check the status register
                      4D      19  0067   169          BLSS    ERROR                   ; Bit 15 set is the error indicator
                0E 54   50    E0  0069   170          BBS     R0,R4,30$               ; Done with this step?
                   51   1B    DB  006D   171          MFPR    #PR$_TODR, R1           ; No, pick up time
                   52   51    D1  0070   172          CMPL    R1,R2                   ; Are we past due time?
                        41    1A  0073   173          BGTRU   ERROR                   ; Yep, error
                        EC    11  0075   174          BRB     LOOP                    ; No, try again
                              0077   175
                              0077   176 BYTE_OFF:
                   FFFFFE00   0077   177          .LONG   ^C<^X1FF>               ; Mask for byte offset in page
                              007B   178
                02 A7   83    B0  007B   179 30$:     MOVW    (R3)+,UDASA(R7)         ; Send the controller the next step
                D8 50   0E    F3  007F   180          AOBLEQ  #S4,R0,TIME             ; Set for next step
                              0083   181 ;
                              0083   182 ; Initialization complete.  Write the packet address in the ring.
                              0083   183 ;
              51   0250'CF    9E  0083   184          MOVAB   W^RSPPKT, R1            ; Get the address of response packet
                51   EC AF   CA  0088   185          BICL    BYTE_OFF, R1
                   51   56    C9  008C    .1         BISL3   R6, R1, W^RD            ; Set byte offset in ring desc
                   0210'CF      008F
              51   021C'CF    9E  0092    .2         MOVAB   W^CMDPKT, R1            ; Get the address of command packet
                   55   51    D0  0097    .3         MOVL    R1,R5                   ; Save pointer
                51   DA AF    CA  009A    .4         BICL    BYTE_OFF, R1
                   51   56    C9  009E    .5         BISL3   R6, R1,W^CD             ; Set byte offset in ring desc
                   0214'CF      00A1
-5                              00A4   191 ;
                              00A4   192 ; Now bring the device on-line
                              00A4   193 ;
                   85   01    D0  00A4   194          MOVL    #1,(R5)+                ; Set command ref number
                85   64 A9    9A  00A7   195          MOVZBL  RPB$W_UNIT(R9),(R5)+    ; Put unit number in cmd packet field
                   85   09    9A  00AB   196          MOVZBL  #MSCP$K_OP_ONLIN,(R5)+  ; Set opcode to bring drive online
                        85    7C  00AE   197          CLRQ    (R5)+                   ; Clear byte count, buff desc
                        65    7C  00B0   198          CLRQ    (R5)                    ;  buff desc and LBN
                     01F7     30  00B2   199          BSBW    IO                      ; Send it out
                              04  00B5   200          RET
                              00B6   201
```

```
                98 58    F5  00B6   202 ERROR:  SOBGTR  R8,RETRY                ; Try once again
           50   0054 8F   3C  00B9   203         MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                          04  00BE   204         RET
                              00BF   205
                              00BF   206         .DISABLE LSB
                  000000C0    00BF   207         .=<.+1>&-2
                  80000000    00C0   208 VALID:  .LONG   ^X80000000             ; SIgn bit set
                              00C4    .1 Ds$Gb_Uda50_Init::                     ; Global label
                              00C4    .2 Init:                                  ; convenient local label
                          00  00C4    .3         .Byte   0                      ; .. init flag
                              00C5   209 ;
                              00C5   210 ; RINGS
                              00C5   211 ;
                              00C5   212 ;
                              C0C5    .1
                              00C5    .2 .Align  Page
                              0200    .3
                      8000    0200   213 INTTBL: .WORD   OWN                    ; Step 1 pattern
                  00000000    0202   214         .LONG   0                      ; Step 2 & 3 pattern
                      0001    0206   215         .WORD   GO
                              0208   216
                 0000 0000    0208   217         .WORD   0,0                    ; Reserved
                      0000    020C   218 CMDINT: .WORD   0                      ; Command status word
                      0000    020E   219 RSPINT: .WORD   0                      ; Response status word
                              0210   220 RING:
                  00000000    0210   221 RD:     .LONG   0                      ; UNIBUS address of response ring
                  00000000    0214   222 CD:     .LONG   0                      ; UNIBUS address of command ring
                              0218   223 ;
                      0030    0218   224         .WORD   48                     ; Length of message
                      0001    021A   225         .WORD   1                      ; ID
                      0001    021C   226 CMDPKT: .WORD   1
                  0000024C    021E   227         .BLKW   23                     ; Full envelope
                              024C   228 ;
                  00000250    024C   229         .BLKW   2
                              0250    .1 UDA_Response::                         ; Global label for CONFIG
                  00000280    0250   230 RSPPKT: .BLKW   24
```

```
                     0280   232          .SBTTL  UDA50 Bootstrap driver QIO
                     0280   233
                     0280   234 ;++
                     0280   235 ;
                     0280   236 ; Inputs:
                     0280   237 ;
                     0280   238 ;       R3      - base address of adapter's register space
                     0280   239 ;       R5      - lbn for current piece of transfer
                     0280   240 ;       R6      - contains 0
                     0280   241 ;       R7      - address of the device's CSR
                     0280   242 ;       R8      - size of transfer in bytes
                     0280   243 ;       R9      - address of the RPB
                     0280   244 ;       R10     - starting address of transfer (byte offset in first
                     0280   245 ;                 page ORed with starting map register number)
                     0280   246 ;       R11     - LBN at start of transfer
                     0280   247 ;
                     0280   248 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                     0280   249 ;       SIZE(AP)- Size of transfer in bytes
                     0280   250 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                     0280   251 ;
                     0280   252 ; Implicit inputs:
                     0280   253 ;
                     0280   254 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                     0280   255 ;
                     0280   256 ; Outputs:
                     0280   257 ;
                     0280   258 ;       R0 - status code
                     0280   259 ;
                     0280   260 ;               SS$_NORMAL      - successful transfer
                     0280   261 ;               SS$_NOSUCHDEV   - unsupported device
                     0280   262 ;               SS$_CTRLERR     - fatal controller error
                     0280   263 ;
                     0280   264 ;       R3 - must be preserved
                     0280   265 ;
                     0280   266 ;
                     0280   267 ;--
                     0280   268
           00000010  0280   269 FUNC = 16
           00000014  0280   270 MODE = 20
                     0280   271
                     0280   272 UD_DRIVER:                                ; UDA50 device driver.
                     0280   273
                     0280   274 ;
                     0280    .1 ; Start out by re-initing the UDA if it hasn't been inited yet,
                     0280    .2 ; or if there was an error previously.
                     0280    .3 ;
                     0280    .4
  0D FE40 CF    E8   0280    .5          BlbS    W^Init, 10$             ; Branch if already set
FD76 CF    00    FB   0285    .6          CallS   #0, Ud_Init             ; Call init code
     51 50    E9   028A    .7          BlbC    R0, Error1              ; Exit if error
FE32 CF    01    90   028D    .8          MovB    #1, W^Init              ; Set init byte
                     0292    .9
                     0292   .10 ;
                     0292   275 ; Translate the I/O function code into a device-dependent function
                     0292   276 ; code for this disk.
                     0292   277 ;
                     0292   278
```

```
                8E AF    21    90  0292     .1 10$:     MOVB    #MSCP$K_OP_READ, -      ; Assume read
-1                                  0296    280                 CMDPKT+MSCP$B_OPCODE
                20    10 AC    D1  0296    281         CMPL    FUNC(AP),#IO$_WRITELBLK ; Check for write function
                         04    12  029A    282         BNEQ    20$                     ; No, do read
                84 AF    22    90  029C    283         MOVB    #MSCP$K_OP_WRITE, -     ; Set write function code
                                    02A0    284                 CMDPKT+MSCP$B_OPCODE
                94 AF    55    D0  02A0    285 20$:    MOVL    R5,CMDPKT+MSCP$L_LBN    ; Set the logical block number
                80 AF    58    D0  02A4    286         MOVL    R8,CMDPKT+MSCP$L_BYTE_CNT ; Set the byte count
                80 AF    5A    D0  02A8    287         MOVL    R10,CMDPKT+MSCP$B_BUFFER; Set the UNIBUS map register
                54    02 A7    B0  02AC    288 IO:     MOVW    UDASA(R7),R4            ; Controller offline?
                         2C    12  02B0    289         BNEQ    ERROR1                  ; Yep, error out
                   8000 8F    A8  02B2    290         BISW    #OWN, CD+2              ; Set controller ownership
                   FF5D CF        02B6
                   8000 8F    A8  02B9    291         BISW    #OWN, RD+2              ; Ditto
                   FF52 CF        02BD
                   54    67   B0  02C0    292         MOVW    UDAIP(R7),R4            ; Tell controller to read
                54    02 A7    B0  02C3    293         MOVW    UDASA(R7),R4            ; Any problems?
                         15    12  02C7    294         BNEQ    ERROR1                  ; Yes
                54    02 A7    B0  02C9     .1 30$:    MOVW    UdaSa(R7), R4           ; Any problems?
                         0F    12  02CD     .2         BNeq    Error1                  ; Yes
                   FF3F CF    B5  02CF     .3         TSTW    RD+2                    ; Any response back?
-1                       F4    19  02D3    296         BLSS    30$                     ; No, spin until there is
                   82 AF    B5  02D5    297         TSTW    RSPPKT+MSCP$W_STATUS    ; Any drive errors?
                         04    12  02D8    298         BNEQ    ERROR1                  ; Yes
                                    02DA     .1
-1                                  02DA    300 ;
                                    02DA    301 ; Transfer is complete. Return with success status code.
                                    02DA    302 ;
                50    01    3C    02DA    303         MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                            05    02DD    304         RSB                             ; AND RETURN
                                    02DE    305 ;
                                    02DE    306 ; Error occured during transfer.  Return and retry.
                                    02DE    307 ;
                                    02DE    308
                                    02DE    309 ERROR1:
                FDE2 CF    94    02DE     .1         ClrB    W^Init                  ; Must re-init next time
50              0054 8F    3C    02E2    310         MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                            05    02E7    311         RSB                             ; Return to BOOTDRIVR
                                    02E8    312
                     000002E8       02E8    313 UD_DRVSIZ=.-START
                                    02E8    314
                                    02E8    315         .END
```

```
$TABLE                          = 00000000 R   D  03
BTD$K_UDA                       = 00000011     D
BYTE_OFF                          00000077 R   D  02
CD                                00000214 R   D  02
CMDINT                            0000020C R   D  02
CMDPKT                            0000021C R   D  02
DS$GB_UDA50_INIT                  000000C4 RG  D  02
ERROR                             000000B6 R   D  02
ERROR1                            000002DE R   D  02
FUNC                            = 00000010     D
GO                              = 00000001     D
INIT                              000000C4 R   D  02
INTTBL                            00000200 R   D  02
IO                                000002AC R   D  02
IO$_WRITELBLK                   = 00000020     D
LOOP                              00000063 R   D  02
MODE                            = 00000014     D
MSCP$B_BUFFER                   = 00000010     D
MSCP$B_OPCODE                   = 00000008     D
MSCP$K_OP_ONLIN                 = 00000009     D
MSCP$K_OP_READ                  = 00000021     D
MSCP$K_OP_WRITE                 = 00000022     D
MSCP$L_BYTE_CNT                 = 0000000C     D
MSCP$L_LBN                      = 0000001C     D
MSCP$W_STATUS                   = 0000000A     D
OWN                             = 00008000     D
PR$_MAPEN                       = 00000038     D
PR$_TODR                        = 0000001B     D
PTE$M_PFN                       = 001FFFFF     D
RD                                00000210 R   D  02
RETRY                             00000051 R   D  02
RING                              00000210 R   D  02
RPB$L_ADPPHY                    = 0000005C     D
RPB$L_ADPVIR                    = 00000060     D
RPB$L_CSRPHY                    = 00000054     D
RPB$L_CSRVIR                    = 00000058     D
RPB$L_SVASPT                    = 00000050     D
RPB$W_UNIT                      = 00000064     D
RSPINT                            0000020E R   D  02
RSPPKT                            00000250 R   D  02
S1                              = 0000000B     D
S4                              = 0000000E     D
SS$_CTRLERR                     = 00000054     D
SS$_NORMAL                      = 00000001     D
START                             00000000 R   D  02
TIME                              0000005B R   D  02
UBA$L_MAP                       = 00000800     D
UDAIP                           = 00000000     D
UDASA                           = 00000002     D
UDA_RESPONSE                      00000250 RG  D  02
UD_DRIVER                         00000280 R   D  02
UD_DRVSIZ                       = 000002E8     D
UD_INIT                           00000000 RG  D  02
UMR                             = 000001E[     D
VA$S_VPN                        = 00000015     D
VA$V_VPN                        = 00000009     D
VALID                             000000C0 R   D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	000002E8 (744.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
BOOTDRIVR_4	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	84.1 (2)	84.1 (2)
BTD$K_UDA	=00000011		84.1 (2)
BYTE_OFF	00000077-R	176 (3)	#-140 (3) #-185 (3) #-185.4 (3)
CD	00000214-R	222 (3)	#-185.5 (3) #-290 (4)
CMDINT	0000020C-R	218 (3)	
CMDPKT	0000021C-R	226 (3)	185.2 (3) #-280 (4) #-284 (4) #-285 (4) #-286 (4) #-287 (4)
DS$GB_UDA50_INIT	000000C4-R	208.1 (3)	
ERROR	00000086-R	202 (3)	#-169 (3) #-173 (3)
ERROR1	000002DE-R	309 (4)	#-274.7 (4) #-289 (4) #-294 (4) #-294.2 (4) #-298 (4)
FUNC	=00000010	269 (4)	#-281 (4)
GO	=00000001	68 (2)	215 (3)
INIT	000000C4-R	208.2 (3)	#-274.5 (4) #-274.8 (4) #-309.1 (4)
INTTBL	00000200-R	213 (3)	139 (3) #-142 (3) 159.1 (3)
IO	000002AC-R	288 (4)	#-199 (3)
IO$_WRITELBLK	=00000020		#-281 (4)
LOOP	00000063-R	168 (3)	#-174 (3)
MODE	=00000014	270 (4)	
MSCP$B_BUFFER	=00000010		#-287 (4)
MSCP$B_OPCODE	=00000008		#-280 (4) #-284 (4)
MSCP$K_OP_ONLIN	=00000009		#-196 (3)
MSCP$K_OP_READ	=00000021		#-278.1 (4)
MSCP$K_OP_WRITE	=00000022		#-283 (4)
MSCP$L_BYTE_CNT	=0000000C		#-286 (4)
MSCP$L_LBN	=0000001C		#-285 (4)
MSCP$W_STATUS	=0000000A		#-297 (4)
OWN	=00008000	69 (2)	213 (3) #-290 (4) #-291 (4)
PR$_MAPEN	=00000038		#-110.5 (3)
PR$_TODR	=0000001B		#-166 (3) #-171 (3)
PTE$M_PFN	=001FFFFF		#-150 (3)
RD	00000210-R	221 (3)	#-185.1 (3) #-291 (4) #-294.3 (4)
RETRY	00000051-R	159.1 (3)	#-202 (3)
RING	00000210-R	220 (3)	#-142 (3)
RPB$L_ADPPHY	=0000005C		144 (3) #-145 (3)
RPB$L_ADPVIR	=00000060		144 (3)
RPB$L_CSRPHY	=00000054		146 (3) #-147 (3)
RPB$L_CSRVIR	=00000058		146 (3)
RPB$L_SVASPT	=00000050		#-149 (3)
RPB$W_UNIT	=00000064		#-195 (3)
RSPINT	0000020E-R	219 (3)	
RSPPKT	00000250-R	230 (3)	184 (3) #-297 (4)
S1	=0000000B	70 (2)	#-161 (3)
S4	=0000000E	71 (2)	#-180 (3)
SS$_CTRLERR	=00000054		#-203 (3) #-310 (4)
SS$_NORMAL	=00000001		#-303 (4)
START	00000000-R	84.6 (2)	313 (4) 84.1 (2)
TIME	0000005B-R	166 (3)	#-180 (3)
UBA$L_MAP	=00000800		151 (3)
UDAIP	=00000000	66 (2)	#-162 (3) #-292 (4)

```
UDASA                           =00000002     67     (2)    #-168     (3)    #-179     (3)    #-288     (4)    #-293     (4)
                                                            #-294.1   (4)
UDA_RESPONSE                     00000250-R   229.1  (3)
UD_DRIVER                        00000280-R   272    (4)      84.1    (2)
UD_DRVSIZ                       =000002E8     313    (4)      84.1    (2)
UD_INIT                          00000000-R   110.1  (3)      274.6   (4)
UMR                             =000001EE     72     (2)    #-138     (3)      151     (3)
VA$S_VPN                        =00000015                   #-143     (3)
VA$V_VPN                        =00000009                   #-143     (3)
VALID                            000000C0-R   208    (3)    #-152     (3)    #-154     (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	82 (2)	82 (2)
$BTDDEF	1	50 (2)	50 (2)
$DEFINI	1	50 (2)	50 (2) 51 (2) 52 (2) 53 (2) 54 (2) 55 (2) 56 (2) 57 (2) 58 (2) 59 (2)
$IODEF	17	51 (2)	51 (2)
$MSCPDEF	18	52 (2)	52 (2)
$PRDEF	4	46.4 (2)	53 (2)
$PTEDEF	3	54 (2)	54 (2)
$RPBDEF	5	55 (2)	55 (2)
$SSDEF	21	56 (2)	56 (2)
$UBADEF	6	57 (2)	57 (2)
$UBIDEF	2	58 (2)	58 (2)
$VADEF	1	59 (2)	59 (2)
ASSUME	1	144 (3)	144 (3) 146 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.10	00:00:00.29
Command processing	147	00:00:00.80	00:00:01.92
Pass 1	605	00:00:17.05	00:00:21.83
Symbol table sort	3	00:00:02.38	00:00:02.64
Pass 2	129	00:00:03.47	00:00:07.89
Symbol table output	9	00:00:00.11	00:00:00.13
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	23	00:00:00.31	00:00:00.70
Assembler run totals	964	00:00:24.25	00:00:35.44

The working set limit was 1000 pages.
76858 bytes (151 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1563 non-local and 5 local symbols.
430 source lines were read in Pass 1, producing 0 object records in Pass 2.
56 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYS0.SYSMAINT]DS.MLB;218	1
DMA1:[SYS0.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	7
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	14

1577 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) PUBTDRIVR/UPDA=(PUBTDRIVR.UPD,PUBTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

```
;    0001  0     %TITLE '*** QA Check Routines'
;    0002  0     MODULE QACHECKS (
;    0003  0               ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE),
;    0004  0               IDENT = '6.5-03'
;    0005  0               ) =
;    0006  0     !
;    0007  0     ! COPYRIGHT (c) 1979, 1981, 1983
;    0008  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0009  0     !
;    0010  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0011  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0012  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0013  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0014  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0015  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0016  0     ! REMAIN IN DEC.
;    0017  0     !
;    0018  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0019  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0020  0     ! CORPORATION.
;    0021  0     !
;    0022  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0023  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0024  0     !
;    0025  0     !++
;    0026  0     ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0027  0     !
;    0028  0     ! ABSTRACT:
;    0029  0     !
;    0030  0     !       This module contains the check routines required to perform QA on diagnostic programs.
;    0031  0     !
;    0032  0     ! AUTHOR:
;    0033  0     !
;    0034  0     !       Jack Stansbury
;    0035  0     !
;    0036  0     ! CREATION DATE:
;    0037  0     !
;    0038  0     !               15-October-1981, Version 6.5-00
;    0039  0     !       00      Put in code for all the support routines plus the following check routines: QA$Normal_Start,
;    0040  0     !               QA$Multiple_Pass, QA$Loop_On_Test, QA$Run_Backwards.
;    0041  0     !
;    0042  0     ! MODIFIED:
;    0043  0     !
;    0044  0     !       01      14-December-1981, Jack Stansbury, Version 6.5-01
;    0045  0     !               Added remaining QA check routines. Separated the check routines (into QACHECKS.B32) and the
;    0046  0     !               support routines (QAMAIN.B32) because the QA.B32 module was getting too large!
;    0047  0     !
;    0048  0     !       02      19-Jan-1983, Jack Stansbury, Version 6.10 (?)
;    0049  0     !               Took out the Error Phase One check because it doesn't work properly with
;    0050  0     !               issuing error messages from subroutines in the diagnostic program.
;    0051  0     !
;    0052  0     !       03      Bob Bergazzi    May 17, 1983    Version 6.11
;    0053  0     !               Changed the order of searching libraries.
;    0054  0     !--
```

```
;    0055  0     %SBTTL  'Table of Contents'
;    0056  1     BEGIN
;    0057  1     !
;    0058  1     ! TABLE OF CONTENTS:
;    0059  1     !
;    0060  1
;    0061  1     FORWARD ROUTINE
;    0062  1             QA$Normal_Start         :       ADDRESSING_MODE (LONG_RELATIVE),
;    0063  1             QA$Multiple_Pass        :       ADDRESSING_MODE (LONG_RELATIVE),
;    0064  1             QA$Loop_On_Test         :       ADDRESSING_MODE (LONG_RELATIVE),
;    0065  1             QA$Loop_On_Subtest      :       ADDRESSING_MODE (LONG_RELATIVE),        !                       [01]
;    0066  1             QA$Run_Backwards        :       ADDRESSING_MODE (LONG_RELATIVE);        !                       [02]
```

```
;    0067  1     %SBTTL  'Libraries'
;    0068  1     !
;    0069  1     ! LIBRARIES:
;    0070  1     !
;    0071  1
;    0072  1     LIBRARY
;    0073  1             '$DIAG';                !                                   [03]
;    0074  1
;    0075  1     LIBRARY
;    0076  1             '$DS';                  !                                   [03]
;    0077  1
;    0078  1     LIBRARY
;    0079  1             'SYS$LIBRARY:LIB';      !                                   [03]
```

```
;    0080  1     %SBTTL  'Included Macros and Literals'
;    0081  1     !
;    0082  1     ! INCLUDED MACROS AND LITERALS
;    0083  1     !
;    0084  1
;    0085  1             $DS_QADEF;                  ! Define the branch codes.
;    0086  1             $DS_DSADEF;                 ! Define the DSA flags.
;    0087  1             DSQA;                       ! Define the DSQA$K_ constants for the routine names.
;    0088  1             DS_QADEFS;                  ! Define the common QA definitions like QA$K_.          [1]
```

```
;    0089  1      %SBTTL  'External Routines'
;    0090  1      !
;    0091  1      ! EXTERNAL ROUTINE REFERENCES:
;    0092  1      !
;    0093  1
;    0094  1      EXTERNAL ROUTINE
;    0095  1              QA$Find_User_Call_Frame :          ADDRESSING_MODE (LONG_RELATIVE),      ! Find the user's last frame.   [01]
;    0096  1              QA$Print_Forced_Errors  : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),      ! Print the Forced-Error table. [01]
;    0097  1              QA$Add_Forced_Error     : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),      ! Add a forced error.           [01]
;    0098  1              QA$Add_QA_Error         : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),      ! Add a QA error.               [01]
;    0099  1
;    0100  1              DSX$Abort;
```

```
;    0101  1     %SBTTL  'External Own Storage'
;    0102  1     !
;    0103  1     ! EXTERNAL OWN STORAGE
;    0104  1     !
;    0105  1
;    0106  1     EXTERNAL
;    0107  1             DS$GL_FSTTEST,                  ! First test to execute
;    0108  1             DS$GL_LSTTEST,                  ! Last test to execute
;    0109  1             DS$GL_SUBTEST,                  ! The subtest to loop on                    [01]
;    0110  1
;    0111  1             DS$GL_CLIBASE,                  ! Base of CLI data table
;    0112  1
;    0113  1             QA$W_TestLoops    : WORD,       ! These are defined in the QAFLAGS module.  [02]
;    0114  1             QA$W_SubtestLoops : WORD,       ! They contain the current settings of the
;    0115  1             QA$W_MultiplePass : WORD,       ! QA flags.
;    0116  1
;    0117  1             !+                                                                          [1]
;    0118  1             ! The following are defined in the QAMAIN.B32 module.                       [1]
;    0119  1             !-                                                                          [1]
;    0120  1
;    0121  1             QA$AOB_Routine_State : BITVECTOR,   !                                       [1]
;    0122  1             QA$AOB_Check_State   : BITVECTOR,   !                                       [1]
;    0123  1             QA$AOB_Flags         : BITVECTOR,   !                                       [1]
;    0124  1
;    0125  1             QA$W_Branch       : WORD,       !                                           [1]
;    0126  1             QA$W_Error_Number : WORD,       !                                           [1]
;    0127  1             QA$W_Save_Test    : SIGNED WORD,! The saved starting test number.           [1]
;    0128  1             QA$W_Save_Last    : SIGNED WORD;! The saved ending test number.             [1]
```

```
;    0129  1     %SBTTL  'Psect Definitions'
;    0130  1     !
;    0131  1     ! PSECT DEFINITIONS:
;    0132  1     !
;    0133  1
;    0134  1     PSECT
;    0135  1             Plit   = Data (NoExecute, Share, NoWrite,
;    0136  1                            Addressing_Mode (Long_Relative));
;    0137  1
;    0138  1     PSECT
;    0139  1             Code   = Code (Execute, Share, NoWrite,
;    0140  1                            Addressing_Mode (Long_Relative));
;    0141  1
;    0142  1     PSECT
;    0143  1             Own    = Work (NoExecute, NoShare, Write,
;    0144  1                            Addressing_Mode (Long_Relative));
;    0145  1
;    0146  1     PSECT
;    0147  1             Global = Work (NoExecute, NoShare, Write,
;    0148  1                            Addressing_Mode (Long_Relative));
```

```
;    0149  1     %SBTTL  'Local Macro Definitions'
;    0150  1     !
;    0151  1     ! LOCAL MACRO DEFINITIONS:
;    0152  1     !
;    0153  1
;    0154  1     MACRO
;    0155  1             !+
;    0156  1             ! Usage:
;    0157  1             !       ASSERT ((.A GTR .B), 'A <= B');
;    0158  1             !
;    0159  1             ! If the specified condition is not true, a call to the Logic_Error macro is made with the specified
;    0160  1             ! message string. ASSERT is turned on whenever the compilation is made with the /VARIANT:n qualifier,
;    0161  1             ! where n <> 0.
;    0162  1             !-
;    0163  1
;  M 0164  1             ASSERT (Condition, Assert_String) =
;  M 0165  1                     %IF %VARIANT
;  M 0166  1                     %THEN
;  M 0167  1                             (
;  M 0168  1                             IF (NOT (Condition)) THEN
;  M 0169  1                                     Logic_Error (Assert_String)
;  M 0170  1                             )
;  M 0171  1                     %ELSE
;  M 0172  1                             ! No code put in.
;  M 0173  1                     %FI
;    0174  1             %;
;    0175  1
;    0176  1     MACRO
;    0177  1             !+
;    0178  1             ! Usage:
;    0179  1             !       Logic_Error ('String to be printed out');
;    0180  1             !
;    0181  1             ! This macro is used whenever a logic error is detected. It is called from ASSERT whenever the assertion
;    0182  1             ! is false. This can be called anywhere else where an error is known to exist and and expression does not
;    0183  1             ! have to be evaluated to determine whether or not an error exists.
;    0184  1             !-
;    0185  1
;  M 0186  1             Logic_Error (Error_String) =
;  M 0187  1                     (
;  M 0188  1                     BIND
;  M 0189  1                             ES = $ASCIC ('!/**QA Internal Error: ', Error_String, '!/');    !                    [1]
;  M 0190  1
;  M 0191  1                     $DS_PRINTF (ES);
;  M 0192  1                     )
;    0193  1             %;
;    0194  1
```

```
:    0195  1     MACRO
:    0196  1             !+
:    0197  1             ! This macro is used only in the debugging stages. It's only parameter is a string giving, e.g., the
:    0198  1             ! name of a routine. DEBUG is turned on whenever the compilation is made with the /VARIANT:n qualifier,
:    0199  1             ! where n <> 0.
:    0200  1             !-
:    0201  1
: M  0202  1             DEBUG (Debug_String) =
: M  0203  1                     %IF %VARIANT
: M  0204  1                     %THEN
: M  0205  1                             (
: M  0206  1                             IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN
: M  0207  1                                     $DS_PRINTF ($ASCIC ('!/', Debug_String, '!/'))
: M  0208  1                             )
: M  0209  1                     %ELSE
: M  0210  1                             ! No code put in.
: M  0211  1                     %FI
:    0212  1             %;
:    0213  1
:    0214  1     MACRO
:    0215  1             !+
:    0216  1             ! Define a macro that takes an expression as an argument. The value of this macro is either
:    0217  1             ! one or zero. All bits but the least significant one are cleared.
:    0218  1             !-
:    0219  1
: M  0220  1             Boolean (Expression) =
:    0221  1                     ((Expression) AND 1) %;
```

```
;    0222  1     %SBTTL  'Module-Global Static Storage'
;    0223  1
;    0224  1     !
;    0225  1     ! MODULE-GLOBAL DECLARATIONS:
;    0226  1     !
;    0227  1
;    0228  1             MODNAM ('QACHECKS');                    ! Define module name                    [1]
```

```
;    0229  1     %SBTTL  'ASCIC Bindings'
;    0230  1
;    0231  1     !+
;    0232  1     ! These are the error messages output by the check routines when they find a QA error.
;    0233  1     !-
;    0234  1
;    0235  1     GLOBAL BIND                                                                                     [01]
;    0236  1             QA$T_Operator_Request_Message =                                                         [01]
;    0237  1                     $ASCIC ('!/**QA ERROR** Program requested operator intervention.!/');
;    0238  1
;    0239  1     BIND                                                                                            [01]
;    0240  1             Cleanup_Error_Message =
;    0241  1                     $ASCIC ('!/**QA ERROR** Error reported in cleanup code!/'),
;    0242  1
;    0243  1             Error_Reported_Message =
;    0244  1                     $ASCIC ('!/**QA ERROR** Error reported!/'),
;    0245  1
;    0246  1             !+
;    0247  1             ! One of these messages in printed AFTER one of the above three is printed.             [02]
;    0248  1             !-
;    0249  1
;    0250  1             Normal_Start_Message =
;    0251  1                     $ASCIC ('**QA ERROR** Normal start check failed!/'),                    !       [02]
;    0252  1
;    0253  1             Multiple_Pass_Message =
;    0254  1                     $ASCIC ('**QA ERROR** Multiple pass check failed!/'),                   !       [02]
;    0255  1
;    0256  1             Infinite_Test_Message =
;    0257  1                     $ASCIC ('**QA ERROR** Infinite loop-on-test check failed!/'),           !       [02]
;    0258  1
;    0259  1             Infinite_Subtest_Message =                                                      !       [01]
;    0260  1                     $ASCIC ('**QA ERROR** Infinite loop-on-subtest check failed!/'),        !       [02]
;    0261  1
;    0262  1             Run_Backwards_Message =
;    0263  1                     $ASCIC ('**QA ERROR** Run individual tests in reverse order check failed!/');   !   [02]
```

```
;    0264  1      !+
;    0265  1      ! These are the header lines that each check routine will print in its own initialization code section.
;    0266  1      ! The first three are defined globally because QA$Dump uses them.
;    0267  1      !-
;    0268  1
;    0269  1      GLOBAL BIND
;    0270  1              QA$T_Header_1 =
;    0271  1                      $ASCIC ('!^!/!/!25** QUALITY ASSURANCE !26**!/'),
;    0272  1
;    0273  1              QA$T_Header_2 =
;    0274  1                      $ASCIC ('!10* !38<PROGRAM: !AC!> REV: !UL.!UL!/'),
;    0275  1
;    0276  1              QA$T_Header_3 =
;    0277  1                      $ASCIC ('!23* !%D!/!/');
```

```
;    0278  1     BIND
;    0279  1
;    0280  1             Header_NS =
;    0281  1                     $ASCIC ('!28* Normal Start!/!/'),
;    0282  1
;    0283  1             Header_MP =
;    0284  1                     $ASCIC ('!28* Multiple Pass!/!/'),
;    0285  1
;    0286  1             Header_LT =
;    0287  1                     $ASCIC ('!24* Infinite Loop-On-Test!/!/'),
;    0288  1                                                                                    [01]
;    0289  1             Header_LS =                                                            [01]
;    0290  1                     $ASCIC ('!22* Infinite Loop-On-Subtest!/!/'),
;    0291  1
;    0292  1             Header_RB =
;    0293  1                     $ASCIC ('!25* Run Tests Backwards!/!/');                       [02]
```

```
;    0294  1      %SBTTL 'QA$Normal_Start'
;    0295  1
;    0296  1      GLOBAL ROUTINE QA$Normal_Start (Hook_Point) =
;    0297  1
;    0298  1      !++
;    0299  1      ! FUNCTIONAL DESCRIPTION:
;    0300  1      !
;    0301  1      !       This routine performs the Normal Start QA check.
;    0302  1      !
;    0303  1      ! CALLER:
;    0304  1      !
;    0305  1      !       QA$Main
;    0306  1      !
;    0307  1      ! FORMAL PARAMETERS:
;    0308  1      !
;    0309  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0310  1      !                    The constants are defined in the DSQA macro.
;    0311  1      !
;    0312  1      ! IMPLICIT INPUTS:
;    0313  1      !
;    0314  1      !       DSA$GL_FLAGS longword.
;    0315  1      !       QA$AOB_Check_State bitvector.
;    0316  1      !
;    0317  1      ! IMPLICIT OUTPUTS:
;    0318  1      !
;    0319  1      !       NONE
;    0320  1      !
;    0321  1      ! COMPLETION CODES:
;    0322  1      !
;    0323  1      !       0 - QA error detected; unsuccessful return
;    0324  1      !       1 - No QA errors; successful return
;    0325  1      !
;    0326  1      ! SIDE EFFECTS:
;    0327  1      !
;    0328  1      !       NONE
;    0329  1      !
;    0330  1      !--
```

```
;    0331  2      BEGIN
;    0332  2              SELECTONE .Hook_Point OF
;    0333  2                      SET
;    0334  2                      [DSQA$K_Start_VRReStart]:
;    0335  3                              BEGIN
;    0336  3                              $DS_PRINTF (QA$T_Header_1);
;    0337  3                              $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0338  3                              $DS_PRINTF (QA$T_Header_3, 0);  ! 0 is for the current time
;    0339  3                              $DS_PRINTF (Header_NS);
;    0340  3
;    0341  3                              QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0342  4                              RETURN (QA$K_Success)
;    0343  2                              END;
;    0344  2
;    0345  2                      [DSQA$K_Error_Error_End]:
;    0346  3                              BEGIN
;    0347  3                              IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN
;    0348  4                                      $DS_PRINTF (Cleanup_Error_Message)
;    0349  3                              ELSE
;    0350  3                                      $DS_PRINTF (Error_Reported_Message);
;    0351  3
;    0352  3                              $DS_PRINTF (Normal_Start_Message);              !                                       [02]
;    0353  3
;    0354  3                              QA$Add_QA_Error (QA$K_Normal_Start);
;    0355  3                              QA$AOB_Routine_State [QA$V_Error_Found] = True;
;    0356  3                              !+
;    0357  3                              ! Return unsucessfully.
;    0358  3                              !-
;    0359  2                              END;
;    0360  2
;    0361  2                      [DSQA$K_Dispat_DSX$EndPass_Mid]:
;    0362  3                              BEGIN
;    0363  3                              DSA$V_TRACE                                     = Off;  ! Turn TRACE bit off.           [01]
;    0364  3                              QA$AOB_Check_State [QA$K_Normal_Start]  = Off;  ! Stop Normal Start check.             [01]
;    0365  3                              QA$AOB_Check_State [QA$K_Multiple_Pass] = On;   ! Start Multiple Pass check.           [01]
;    0366  3                              QA$AOB_Routine_State [QA$V_Check_Done]  = True; ! Indicate end of this check.          [01]
;    0367  4                              RETURN (QA$K_Success)
;    0368  4
;    0369  2                              END;
```

```
;    0370  2                    !+
;    0371  2                    ! The following are not used in this check. So, just return Success for them.
;    0372  2                    !-
;    0373  2
;    0374  2                    [DSQA$K_Dispat_Before_Init,
;    0375  2                     DSQA$K_Dispat_After_Init,
;    0376  2                     DSQA$K_Dispat_CallTest,
;    0377  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,          !                                    [01]
;    0378  2                     DSQA$K_Dispat_DSX$EndPass_End,          !                                    [01]
;    0379  2                     DSQA$K_Dispat_Done_Tests,               !                                    [01]
;    0380  2
;    0381  2                     DSQA$K_Error_Error_Bgn,                 !                                    [01]
;    0382  2
;    0383  2                     DSQA$K_Loop_DSX$BgnSub,
;    0384  2                     DSQA$K_Loop_DSX$EndSub,
;    0385  2                     DSQA$K_Loop_DSX$CkLoop,
;    0386  2
;    0387  2                     DSQA$K_QA_DSX$Branch]:
;    0388  3                             BEGIN
;    0389  4                             RETURN (QA$K_Success)
;    0390  2                             END;
;    0391  2
;    0392  2                    [OTHERWISE]:
;    0393  3                             BEGIN
;    0394  4                             Logic_Error ('Normal Start, Illegal Hook_Point')
;    0395  2                             END;
;    0396  2                    TES;
;    0397  2
;    0398  2                !+
;    0399  2                ! Come to here only if a QA error was found. Return QA$K_Failure to QA$Main, which then takes
;    0400  2                ! appropriate action.
;    0401  2                !-
;    0402  2
;    0403  3                RETURN (QA$K_Failure)
;    0404  1        END;


                                                                .TITLE  QACHECKS *** QA Check Routines
                                                                .IDENT  \6.5-03\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                53 4B 43 45 48 43 41 51 08  00000 P.AAA:  .ASCII  <8>\QACHECKS\
2A 2A 52 4F 52 52 45 20 41 51 2A 2A 2F 21 39  00009 P.AAB:  .ASCII  \9!/**QA ERROR** Program requested operat\
73 65 75 71 65 72 20 6D 41 72 67 6F 72 50 20  00018
                  74 61 72 [ 70 6F 20 64 65 74  00027
6E 6F 69 74 6E 65 76 72 [  74 6E 69 20 72 6F  00031          .ASCII  \or intervention.!/\
                                 2F 21 2E     00040
2A 2A 52 4F 52 52 45 20 41 51 2A 2A 2F 21 2F  00043 P.AAC:  .ASCII  \/!/**QA ERROR** Error reported in cleanu\
64 65 74 72 6F 70 65 72 20 72 6F 72 72 45 20  00052
                  75 6E 61 65 6C 63 20 6E 69 20  00061
                     2F 21 65 64 6F 63 20 70     0006B          .ASCII  \p code!/\
2A 2A 52 4F 52 52 45 20 41 51 2A 2A 2F 21 1F  00073 P.AAD:  .ASCII  <31>\!/**QA ERROR** Error reported!/\
64 65 74 72 6F 70 65 72 20 72 6F 72 72 45 20  00082
                                 2F 21        0009?
4E 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 28  00093 P.AAE:  .ASCII  \(**QA ERROR** Normal start check failed!\
65 68 63 20 74 72 61 74 73 20 6C 61 6D 72 6F  000A2
```

```
                                21 64 65 6C 69 61 66 20 6B 63 000B1
                                                           2F 000BB          .ASCII  \/\
4D 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 29 000BC P.AAF:  .ASCII  \)**QA ERROR** Multiple pass check failed\
68 63 20 73 73 61 70 20 65 6C 70 69 74 6C 75 000CB
                   64 65 6C 69 61 66 20 6B 63 65 000DA
                                              2F 21 000E4          .ASCII  \!/\
49 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 31 000E6 P.AAG:  .ASCII  \1**QA ERROR** Infinite loop-on-test chec\
6E 6F 2D 70 6F 6F 6C 20 65 74 69 6E 69 66 6E 000F5
                   63 65 68 63 20 74 73 65 74 2D 00104
                   2F 21 64 65 6C 69 61 66 20 6B 0010E          .ASCII  \k failed!/\
49 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 34 00118 P.AAH:  .ASCII  \4**QA ERROR** Infinite loop-on-subtest c\
6E 6F 2D 70 6F 6F 6C 20 65 74 69 6E 69 66 6E 00127
                   63 20 74 73 65 74 62 75 73 2D 00136
            2F 21 64 65 6C 69 61 66 20 6B 63 65 68 00140          .ASCII  \heck failed!/\
52 20 2A 2A 52 4F 52 52 45 2C 41 51 2A 2A 41 0014D P.AAI:  .ASCII  \A**QA ERROR** Run individual tests in re\
74 20 6C 61 75 64 69 76 69 64 6E 69 20 6E 75 0015C
                   65 72 20 6E 69 20 73 74 73 65 0016B
65 68 63 20 72 65 64 72 6F 20 65 73 72 65 76 00175          .ASCII  \verse order check failed!/\
                2F 21 64 65 6C 69 61 66 20 6B 63 00184
55 51 20 2A 2A 35 32 21 2F 21 2F 21 5E 21 25 0018F P.AAJ:  .ASCII  \%!^!/!/!25** QUALITY ASSURANCE !26**!/\
45 43 4E 41 52 55 53 53 41 20 59 54 49 4C 41 0019E
                      2F 21 2A 2A 36 32 21 20 001AD
52 47 4F 52 50 3C 38 33 21 20 2A 30 31 21 26 001B5 P.AAK:  .ASCII  \&!10* !38<PROGRAM: !AC!> REV: !UL.!UL!/\
20 3A 56 45 52 20 3E 21 43 41 21 20 3A 4D 41 001C4
                   2F 21 4C 55 21 2E 4C 55 21 001D3
            2F 21 2F 21 44 25 21 20 2A 33 32 21 0C 001DC P.AAL:  .ASCII  <12>\!23* !%D!/!/\
74 53 20 6C 61 6D 72 6F 4E 20 2A 38 32 21 15 001E9 P.AAM:  .ASCII  <21>\!28* Normal Start!/!/\
                         2F 21 2F 21 74 72 61 00 1F8
20 65 6C 70 69 74 6C 75 4D 20 2A 38 32 21 16 001FF P.AAN:  .ASCII  <22>\!28* Multiple Pass!/!/\
                      2F 21 2F 21 73 73 61 50 0020E
20 65 74 69 6E 69 66 6E 49 20 2A 34 32 21 1E 00216 P.AAO:  .ASCII  <30>\!24* Infinite Loop-On-Test!/!/\
21 2F 21 2F 73 65 54 2D 6E 4F 2D 70 6F 6F 4C 00225
                                              2F 00234
20 65 74 69 6E 69 66 6E 49 20 2A 32 32 21 21 00235 P.AAP:  .ASCII  \!!22* Infinite Loop-On-Subtest!/!/\
74 73 65 74 62 75 53 2D 6E 4F 2D 70 6F 6F 4C 00244
                         2F 21 2F 21 00253
73 74 73 65 54 20 6E 75 52 20 2A 35 32 21 1C 00257 P.AAQ:  .ASCII  <28>\!25* Run Tests Backwards!/!/\
   2F 21 2F 21 73 64 72 61 77 6B 63 61 42 20 00266
61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 39 00274 P.AAR:  .ASCII  \9!/**QA Internal Error: Normal Start, Il\
6C 61 6D 72 6F 4E 20 3A 72 6F 72 72 45 20 6C 00283
                   6C 49 20 2C 74 72 61 74 53 20 00292
6E 69 6F 50 5F 6B 6F 6F 48 20 6C 61 67 65 6C 0029C          .ASCII  \legal Hook_Point!/\
                                              2F 21 74 002AB
```

```
DSA$AL_APTMAIL=      65024
DSA$GL_FLAGS=        65024
DSA$GL_APTCOM=       65028
DSA$GL_PASSES=       65032
DSA$GL_UNITS=        65036
DSA$GL_SECTNO=       65040
DSA$GL_SID=          65044
DSA$GL_MSGTYP=       65088
DSA$GL_ERRNO=        65092
DSA$GL_EVENT=        65096
DSA$GL_SUBTNO=       65100
DSA$GL_TESTNO=       65104
DSA$GL_PASSNO=       65108
```

```
                                                                DSA$GL_DEVLEN=      65112
                                                                DSA$GL_DEVNAM=      65116
                                                                DSA$GL_MSGPTR=      65128
                                                                $MODULE=            P.AAA
                                                                QA$T_OPERATOR_REQUEST_MESSAGE==
                                                                                    P.AAB
                                                                CLEANUP_ERROR_MESSAGE=
                                                                                    P.AAC
                                                                ERROR_REPORTED_MESSAGE=
                                                                                    P.AAD
                                                                NORMAL_START_MESSAGE=
                                                                                    P.AAE
                                                                MULTIPLE_PASS_MESSAGE=
                                                                                    P.AAF
                                                                INFINITE_TEST_MESSAGE=
                                                                                    P.AAG
                                                                INFINITE_SUBTEST_MESSAGE=
                                                                                    P.AAH
                                                                RUN_BACKWARDS_MESSAGE=
                                                                                    P.AAI
                                                                QA$T_HEADER_1==     P.AAJ
                                                                QA$T_HEADER_2==     P.AAK
                                                                QA$T_HEADER_3==     P.AAL
                                                                HEADER_NS=          P.AAM
                                                                HEADER_MP=          P.AAN
                                                                HEADER_LT=          P.AAO
                                                                HEADER_LS=          P.AAP
                                                                HEADER_RB=          P.AAQ
                                                                ES=                 P.AAR
                                                                        .EXTRN  QA$FIND_USER_CALL_FRAME
                                                                        .EXTRN  QA$PRINT_FORCED_ERRORS
                                                                        .EXTRN  QA$ADD_FORCED_ERROR
                                                                        .EXTRN  QA$ADD_QA_ERROR
                                                                        .EXTRN  DSX$ABORT, DS$GL_FSTTEST
                                                                        .EXTRN  DS$GL_LSTTEST, DS$GL_SUBTEST
                                                                        .EXTRN  DS$GL_CLIBASE, QA$W_TESTLOOPS
                                                                        .EXTRN  QA$W_SUBTESTLOOPS
                                                                        .EXTRN  QA$W_MULTIPLEPASS
                                                                        .EXTRN  QA$AOB_ROUTINE_STATE
                                                                        .EXTRN  QA$AOB_CHECK_STATE
                                                                        .EXTRN  QA$AOB_FLAGS, QA$W_BRANCH
                                                                        .EXTRN  QA$W_ERROR_NUMBER
                                                                        .EXTRN  QA$W_SAVE_TEST, QA$W_SAVE_LAST
                                                                        .EXTRN  DS$PRINTF

                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                    007C 00000          .ENTRY  QA$NORMAL_START, Save R2,R3,R4,R5,R6            ; 0296
56 00000000G                                EF  9E 00002        MOVAB   QA$AOB_CHECK_STATE, R6                          ;
55 00000000G                                EF  9E 00009        MOVAB   QA$AOB_ROUTINE_STATE, R5                        ;
54 00000000G                                9F  9E 00010        MOVAB   a#DS$PRINTF, R4                                 ;
53 00000000'                                EF  9E 00017        MOVAB   QA$T_HEADER_1, R3                               ;
52          04                              AC  D0 0001E        MOVL    HOOK_POINT, R2                                  ; 0332
12                                          52  D1 00022        CMPL    R2, #18                                         ; 0334
                                            2B  12 00025        BNEQ    1$                                              ;
                                            53  DD 00027        PUSHL   R3                                              ; 0336
64                                          01  FB 00029        CALLS   #1, DS$PRINTF                                   ;
```

```
                                 7E 0000020C   9F   7D 0002C          MOVQ    @#^X0000020C, -(SP)                     ; 0337
                                    00000208   9F   DD 00033          PUSHL   @#^X00000208                            ;
                                           26   A3   9F 00039          PUSHAB  QA$T_HEADER_2                           ;
                                 64             04   FB 0003C          CALLS   #4, DS$PRINTF                           ;
                                                7E   D4 0003F          CLRL    -(SP)                                   ; 0338
                                           4D   A3   9F 00041          PUSHAB  QA$T_HEADER_3                           ;
                                 64             02   FB 00044          CALLS   #2, DS$PRINTF                           ;
                                           5A   A3   9F 00047          PUSHAB  HEADER_NS                               ; 0339
                                 64             01   FB 0004A          CALLS   #1, DS$PRINTF                           ;
                                 65             01   88 0004D          BISB2   #1, QA$AOB_ROUTINE_STATE                ; 0341
                                                6B   11 00050          BRB     9$                                      ; 0342
                                 05             52   D1 00052 1$:      CMPL    R2, #5                                  ; 0345
                                                26   12 00055          BNEQ    4$                                      ;
                              06 65             02   E1 00057          BBC     #2, QA$AOB_ROUTINE_STATE, 2$            ; 0347
                                       FEB4     C3   9F 0005B          PUSHAB  CLEANUP_ERROR_MESSAGE                   ; 0348
                                                04   11 0005F          BRB     3$                                      ;
                                       FEE4     C3   9F 00061 2$:      PUSHAB  ERROR_REPORTED_MESSAGE                  ; 0350
                                 64             01   FB 00065 3$:      CALLS   #1, DS$PRINTF                           ;
                                       FF04     C3   9F 00068          PUSHAB  NORMAL_START_MESSAGE                    ; 0352
                                 64             01   FB 0006C          CALLS   #1, DS$PRINTF                           ;
                                                7E   D4 0006F          CLRL    -(SP)                                   ; 0354
                        00000000G EF             01   FB 00071          CALLS   #1, QA$ADD_QA_ERROR                     ;
                                 65             02   88 00078          BISB2   #2, QA$AOB_ROUTINE_STATE                ; 0355
                                                48   11 0007B          BRB     11$                                     ; 0332
                                                52   D5 0007D 4$:      TSTL    R2                                      ; 0361
                                                12   12 0007F          BNEQ    5$                                      ;
                        0000FE01 9F             04   8A 00081          BICB2   #4, @#^X0000FE01                        ; 0363
                                 66             01   8A 00088          BICB2   #1, QA$AOB_CHECK_STATE                  ; 0364
                                 66             02   88 0008B          BISB2   #2, QA$AOB_CHECK_STATE                  ; 0365
                                 65             08   88 0008E          BISB2   #8, QA$AOB_ROUTINE_STATE                ; 0366
                                                2A   11 00091          BRB     9$                                      ; 0367
                                                05   15 00093 5$:      BLEQ    6$                                      ; 0374
                                 02             52   D1 00095          CMPL    R2, #2                                  ;
                                                23   15 00098          BLEQ    9$                                      ;
                                 06             52   D1 0009A 6$:      CMPL    R2, #6                                  ;
                                                1E   13 0009D          BEQL    9$                                      ;
                                 09             52   D1 0009F          CMPL    R2, #9                                  ;
                                                05   19 000A2          BLSS    7$                                      ;
                                 0B             52   D1 000A4          CMPL    R2, #11                                 ;
                                                14   15 000A7          BLEQ    9$                                      ;
                                 11             52   D1 000A9 7$:      CMPL    R2, #17                                 ;
                                                0F   13 000AC          BEQL    9$                                      ;
                                 13             52   D1 000AE          CMPL    R2, #19                                 ;
                                                05   19 000B1          BLSS    8$                                      ;
                                 15             52   D1 000B3          CMPL    R2, #21                                 ;
                                                05   15 000B6          BLEQ    9$                                      ;
                                 18             52   D1 000B8 8$:      CMPL    R2, #24                                 ;
                                                04   12 000BB          BNEQ    10$                                     ;
                                 50             01   D0 000BD 9$:      MOVL    #1, R0                                  ; 0389
                                                04   000C0            RET                                             ;
                                       00E5     C3   9F 000C1 10$:     PUSHAB  ES                                      ; 0394
                                 64             01   FB 000C5          CALLS   #1, DS$PRINTF                           ;
                                                50   D4 000C8 11$:     CLRL    R0                                      ; 0403
                                                04   000CA            RET                                             ; 0404

; Routine Size:  203 bytes,    Routine Base:  CODE + 0000
```

```
;    0405  1
```

```
;    0406  1      %SBTTL 'QA$Multiple_Pass'
;    0407  1
;    0408  1      GLOBAL ROUTINE QA$Multiple_Pass (Hook_Point) =
;    0409  1
;    0410  1      !++
;    0411  1      ! FUNCTIONAL DESCRIPTION:
;    0412  1      !
;    0413  1      !       This routine performs the QA Multiple Pass check.
;    0414  1      !
;    0415  1      ! CALLER(S):
;    0416  1      !
;    0417  1      !       QA$Main
;    0418  1      !
;    0419  1      ! FORMAL PARAMETERS:
;    0420  1      !
;    0421  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0422  1      !                    The constants are defined in the DSQA macro.
;    0423  1      !
;    0424  1      ! IMPLICIT INPUTS:
;    0425  1      !
;    0426  1      !       NONE
;    0427  1      !
;    0428  1      ! IMPLICIT OUTPUTS:
;    0429  1      !
;    0430  1      !       NONE
;    0431  1      !
;    0432  1      ! COMPLETION CODES:
;    0433  1      !
;    0434  1      !       0 - QA error detected; unsuccessful return
;    0435  1      !       1 - No QA errors; successful return
;    0436  1      !
;    0437  1      ! SIDE EFFECTS:
;    0438  1      !
;    0439  1      !       NONE
;    0440  1      !
;    0441  1      !--
```

```
;    0442  2     BEGIN
;    0443  2             SELECTONE .Hook_Point OF
;    0444  2                     SET
;    0445  2                     [DSQA$K_Start_VRReStart]:                !                                                  [01]
;    0446  3                             BEGIN                                                                              [01]
;    0447  3                             MAP                              !                                                 [01]
;    0448  3                                     DS$GL_CLIBASE : BLOCK [, BYTE]; ! Access the CLI date vector.              [01]
;    0449  3
;    0450  3                             $DS_PRINTF (QA$T_Header_1);
;    0451  3                             $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0452  3                             $DS_PRINTF (QA$T_Header_3, 0);
;    0453  3                             $DS_PRINTF (Header_MP);
;    0454  3
;    0455  3                             DS$GL_CLIBASE [CLI$L_PASS] = .QA$W_MultiplePass;! Set number of passes to requested amt.[01]
;    0456  3                             QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0457  4                             RETURN (QA$K_Success)
;    0458  3                             END;
;    0459  3
;    0460  3                     [DSQA$K_Error_Error_End]:
;    0461  3                             BEGIN
;    0462  3                             IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                          [01]
;    0463  3                                     $DS_PRINTF (Cleanup_Error_Message)              !                          [01]
;    0464  3                             ELSE                                                    !                          [01]
;    0465  3                                     $DS_PRINTF (Error_Reported_Message);            !                          [01]
;    0466  3
;    0467  3                             $DS_PRINTF (Multiple_Pass_Message);             !                                  [02]
;    0468  3
;    0469  3                             QA$Add_QA_Error (QA$K_Multiple_Pass);
;    0470  3                             QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0471  3                             !+
;    0472  3                             ! Return unsucessfully.
;    0473  3                             !-
;    0474  3                             END;
;    0475  3
;    0476  2                     [DSQA$K_Dispat_DSX$EndPass_Mid]:        !                                                  [01]
;    0477  3                             BEGIN
;    0478  3                             !+                                                                                 [01]
;    0479  3                             ! All done the Multiple Pass check. Clean up.                                      [01]
;    0480  3                             !-                                                                                 [01]
;    0481  3
;    0482  3                             QA$AOB_Check_State [QA$K_Multiple_Pass] = Off;  ! Stop Multiple Pass check.        [01]
;    0483  3                             QA$AOB_Check_State [QA$K_Loop_On_Test]  = On;   ! Start Loop on Test check.       [01]
;    0484  3                             QA$AOB_Routine_State [QA$V_Check_Done]  = True; ! Indicate check is done.         [01]
;    0485  4                             RETURN (QA$K_Success)                           !                                  [01]
;    0486  2                             END;
```

```
;    0487  2                    !+
;    0488  2                    ! The following are not used in this check.
;    0489  2                    !-
;    0490  2
;    0491  2                    [DSQA$K_Dispat_Before_Init,
;    0492  2                     DSQA$K_Dispat_After_Init,              !                                                [01]
;    0493  2                     DSQA$K_Dispat_CallTest,
;    0494  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,         !                                                [01]
;    0495  2                     DSQA$K_Dispat_DSX$EndPass_End,         !                                                [01]
;    0496  2                     DSQA$K_Dispat_Done_Tests,              !                                                [01]
;    0497  2
;    0498  2                     DSQA$K_Error_Error_Bgn,                !                                                [01]
;    0499  2
;    0500  2                     DSQA$K_Loop_DSX$BgnSub,
;    0501  2                     DSQA$K_Loop_DSX$EndSub,
;    0502  2                     DSQA$K_Loop_DSX$CkLoop,
;    0503  2
;    0504  2                     DSQA$K_QA_DSX$Branch]:
;    0505  3                             BEGIN
;    0506  4                             RETURN (QA$K_Success)
;    0507  2                             END;
;    0508  2
;    0509  2                    [OTHERWISE]:
;    0510  3                             BEGIN
;    0511  4                             Logic_Error ('Multiple Pass, Illegal Hook_Point')
;    0512  2                             END;
;    0513  2                    TES;
;    0514  3                RETURN (QA$K_Failure)
;    0515  1        END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 3A 002AE P.AAS:    .ASCII  \:!/**QA Internal Error: Multiple Pass, I\  ;
70 69 74 6C 75 4D 20 3A 72 6F 72 72 45 20 6C 002BD                                                              ;
                  49 20 2C 73 73 61 50 20 65 6C 002CC                                                           ;
69 6F 50 5F 6B 6F 6F 48 20 6C 61 67 65 6C 6C 002D6          .ASCII  \llegal Hook_Point!/\                       ;
                              2F 21 74 6E 002E5                                                                 ;

                                                        ES=                 P.AAS

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                  007C 00000    .ENTRY  QA$MULTIPLE_PASS, Save R2,R3,R4,R5,R6       ; 0408
                                56 00000000G   EF   9E 00002    MOVAB   QA$AOB_CHECK_STATE, R6                      ;
                                55 00000000G   EF   9E 00009    MOVAB   QA$AOB_ROUTINE_STATE, R5                    ;
                                54 00000000G   9F   9E 00010    MOVAB   a#DS$PRINTF, R4                             ;
                                53 00000000'   EF   9E 00017    MOVAB   QA$T_HEADER_1, R3                           ;
                                52       04    AC   D0 0001E    MOVL    HOOK_POINT, R2                              ; 0443
                                12             52   D1 00022    CMPL    R2, #18                                     ; 0445
                                               36   12 00025    BNEQ    1$                                          ;
                                               53   DD 00027    PUSHL   R3                                          ; 0450
                                64             01   FB 00029    CALLS   #1, DS$PRINTF                               ;
                                7E 0000020C    9F   7D 0002C    MOVQ    a#^X0000020C, -(SP)                         ; 0451
                                   00000208    9F   DD 00033    PUSHL   a#^X00000208                                ;
```

```
                                    26   A3  9F 00039          PUSHAB  QA$T_HEADER_2                                          ;
                                64       04  FB 0003C          CALLS   #4, DS$PRINTF                                        ;
                                         7E  D4 0003F          CLRL    -(SP)                                                ; 0452
                                    4D   A3  9F 00041          PUSHAB  QA$T_HEADER_3                                        ;
                                64       02  FB 00044          CALLS   #2, DS$PRINTF                                        ;
                                    70   A3  9F 00047          PUSHAB  HEADER_MP                                            ; 0453
                                64       01  FB 0004A          CALLS   #1, DS$PRINTF                                        ;
                00000000G       EF  00000000G  EF  3C 0004D    MOVZWL  QA$W_MULTIPLEPASS, DS$GL_CLIBASE+44                  ; 0455
                                65       01  88 00058          BISB2   #1, QA$AOB_ROUTINE_STATE                             ; 0456
                                         64  11 0005B          BRB     9$                                                   ; 0457
                                05       52  D1 0005D 1$:      CMPL    R2, #5                                               ; 0460
                                         26  12 00060          BNEQ    4$                                                   ;
                          06            65  02  E1 00062      BBC     #2, QA$AOB_ROUTINE_STATE, 2$                         ; 0462
                                    FE84     C3  9F 00066      PUSHAB  CLEANUP_ERROR_MESSAGE                                ; 0463
                                             04  11 0006A      BRB     3$                                                   ;
                                    FEE4     C3  9F 0006C 2$:  PUSHAB  ERROR_REPORTED_MESSAGE                               ; 0465
                                64           01  FB 00070 3$:  CALLS   #1, DS$PRINTF                                        ;
                                    FF2D     C3  9F 00073      PUSHAB  MULTIPLE_PASS_MESSAGE                                ; 0467
                                64           01  FB 00077      CALLS   #1, DS$PRINTF                                        ;
                                             01  DD 0007A      PUSHL   #1                                                   ; 0469
                00000000G       EF           01  FB 0007C      CALLS   #1, QA$ADD_QA_ERROR                                  ;
                                65           02  88 00083      BISB2   #2, QA$AOB_ROUTINE_STATE                             ; 0470
                                             44  11 00086      BRB     11$                                                  ;
                                             52  D5 00088 4$:  TSTL    R2                                                   ; 0476
                                             0B  12 0008A      BNEQ    5$                                                   ;
                                66           02  8A 0008C      BICB2   #2, QA$AOB_CHECK_STATE                               ; 0482
                                66           04  88 0008F      BISB2   #4, QA$AOB_CHECK_STATE                               ; 0483
                                65           08  88 00092      BISB2   #8, QA$AOB_ROUTINE_STATE                             ; 0484
                                             2A  11 00095      BRB     9$                                                   ; 0485
                                             05  15 00097 5$:  BLEQ    6$                                                   ; 0491
                                02           52  D1 00099      CMPL    R2, #2                                               ;
                                             23  15 0009C      BLEQ    9$                                                   ;
                                06           52  D1 0009E 6$:  CMPL    R2, #6                                               ;
                                             1E  13 000A1      BEQL    9$                                                   ;
                                09           52  D1 000A3      CMPL    R2, #9                                               ;
                                             05  19 000A6      BLSS    7$                                                   ;
                                0B           52  D1 000A8      CMPL    R2, #11                                              ;
                                             14  15 000AB      BLEQ    9$                                                   ;
                                11           52  D1 000AD 7$:  CMPL    R2, #17                                              ;
                                             0F  13 000B0      BEQL    9$                                                   ;
                                13           52  D1 000B2      CMPL    R2, #19                                              ;
                                             05  19 000B5      BLSS    8$                                                   ;
                                15           52  D1 000B7      CMPL    R2, #21                                              ;
                                             05  15 000BA      BLEQ    9$                                                   ;
                                18           52  D1 000BC 8$:  CMPL    R2, #24                                              ;
                                             04  12 000BF      BNEQ    10$                                                  ;
                                50           01  D0 000C1 9$:  MOVL    #1, R0                                               ; 0506
                                             04  04 000C4      RET                                                          ;
                                    011F     C3  9F 000C5 10$: PUSHAB  ES                                                   ; 0511
                                64           01  FB 000C9      CALLS   #1, DS$PRINTF                                        ;
                                             50  D4 000CC 11$: CLRL    R0                                                   ; 0514
                                             04  04 000CE      RET                                                          ; 0515

; Routine Size:  207 bytes,    Routine Base:  CODE + 00CB
```

```
;    0516  1      %SBTTL 'QA$Loop_On_Test'
;    0517  1
;    0518  1      GLOBAL ROUTINE QA$Loop_On_Test (Hook_Point) =
;    0519  1
;    0520  1      !++
;    0521  1      ! FUNCTIONAL DESCRIPTION:
;    0522  1      !
;    0523  1      !       This routine performs the QA Loop on Test check.
;    0524  1      !
;    0525  1      ! CALLER(S):
;    0526  1      !
;    0527  1      !       QA$Main
;    0528  1      !
;    0529  1      ! FORMAL PARAMETERS:
;    0530  1      !
;    0531  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0532  1      !                    The constants are defined in the DSQA macro.
;    0533  1      !
;    0534  1      ! IMPLICIT INPUTS:
;    0535  1      !
;    0536  1      !       NONE
;    0537  1      !
;    0538  1      ! IMPLICIT OUTPUTS:
;    0539  1      !
;    0540  1      !       NONE
;    0541  1      !
;    0542  1      ! COMPLETION CODES:
;    0543  1      !
;    0544  1      !       0 - QA error detected; unsuccessful return
;    0545  1      !       1 - No QA errors; successful return
;    0546  1      !
;    0547  1      ! SIDE EFFECTS:
;    0548  1      !
;    0549  1      !       NONE
;    0550  1      !
;    0551  1      !--
```

```
;    0552  2      BEGIN
;    0553  2              OWN
;    0554  2                      Loop_Counter : WORD;
;    0555  2
;    0556  2              SELECTONE .Hook_Point OF
;    0557  2                      SET
;    0558  2                      [DSQA$K_Start_VRReStart]:
;    0559  3                              BEGIN
;    0560  3                              MAP                                     !                                       [01]
;    0561  3                                      DS$GL_CLIBASE : BLOCK [,BYTE]; ! Access the CLI data vector.           [01]
;    0562  3
;    0563  3                              !+
;    0564  3                              ! The initializations required for this check and that are specific to this
;    0565  3                              ! check must be done here.
;    0566  3                              !+
;    0567  3
;    0568  3                              Loop_Counter = 1;                       ! Loop counter for each test.          [01]
;    0569  3                              QA$AOB_Flags [QA$K_Loop_Test_Done] = True;
;    0570  3
;    0571  3                              $DS_PRINTF (QA$T_Header_1);
;    0572  3                              $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0573  3                              $DS_PRINTF (QA$T_Header_3, 0);
;    0574  3                              $DS_PRINTF (Header_LT);
;    0575  3
;    0576  3                              DS$GL_CLIBASE [CLI$L_PASS] = 1; ! Reset to one pass after the Multiple Pass check.  [01]
;    0577  3                              QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0578  4                              RETURN (QA$K_Success)
;    0579  2                              END;
;    0580  2
;    0581  2                      [DSQA$K_Error_Error_End]:
;    0582  3                              BEGIN
;    0583  3                              IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                       [01]
;    0584  4                                      $DS_PRINTF (Cleanup_Error_Message)              !                       [01]
;    0585  3                              ELSE                                                    !                       [01]
;    0586  3                                      $DS_PRINTF (Error_Reported_Message);            !                       [01]
;    0587  3
;    0588  3                              $DS_PRINTF (Infinite_Test_Message);                     !                       [02]
;    0589  3
;    0590  3                              QA$Add_QA_Error (QA$K_Loop_On_Test);
;    0591  3                              QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0592  2                              END;
```

```
;    0593  2                    [DSQA$K_Dispat_DSX$EndPass_Mid]:        !                                                    [01]
;    0594  3                            BEGIN
;    0595  3                            !+
;    0596  3                            ! All done the Loop on Test check. Clean up.                                     [01]
;    0597  3                            !-
;    0598  3
;    0599  3                            QA$AOB_Check_State [QA$K_Loop_On_Test]    = Off;
;    0600  3                            QA$AOB_Check_State [QA$K_Loop_On_Subtest] = On;
;    0601  3                            QA$AOB_Routine_State [QA$V_Check_Done]    = True;
;    0602  3                            QA$AOB_Flags [QA$K_First_Time]            = True;       ! For the next check (LS).   [01]
;    0603  4                            RETURN (QA$K_Success)
;    0604  2                            END;
;    0605  2
;    0606  2                    [DSQA$K_Dispat_CallTest]:
;    0607  3                            BEGIN
;    0608  3                            IF (.Loop_Counter EQL .QA$W_TestLoops) THEN
;    0609  4                                    BEGIN
;    0610  4                                    Loop_Counter = 1;
;    0611  4                                    QA$AOB_Flags [QA$K_Loop_Test_Done] = True
;    0612  4                                    END
;    0613  3                            ELSE
;    0614  4                                    BEGIN
;    0615  4                                    Loop_Counter = .Loop_Counter + 1;
;    0616  4                                    QA$AOB_Flags [QA$K_Loop_Test_Done] = False
;    0617  3                                    END;
;    0618  4                            RETURN (QA$K_Success)
;    0619  2                            END;
```

```
;    0620  2                    !+
;    0621  2                    ! The following are not used in this check.
;    0622  2                    !-
;    0623  2
;    0624  2                    [DSQA$K_Dispat_Before_Init,
;    0625  2                     DSQA$K_Dispat_After_Init,                !                                          [01]
;    0626  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,           !                                          [01]
;    0627  2                     DSQA$K_Dispat_DSX$EndPass_End,           !                                          [01]
;    0628  2                     DSQA$K_Dispat_Done_Tests,                !                                          [01]
;    0629  2
;    0630  2                     DSQA$K_Error_Error_Bgn,                  !                                          [01]
;    0631  2
;    0632  2                     DSQA$K_Loop_DSX$BgnSub,
;    0633  2                     DSQA$K_Loop_DSX$EndSub,
;    0634  2                     DSQA$K_Loop_DSX$CkLoop,
;    0635  2
;    0636  2                     DSQA$K_QA_DSX$Branch]:
;    0637  3                             BEGIN
;    0638  4                             RETURN (QA$K_Success)
;    0639  2                             END;
;    0640  2
;    0641  2                    [OTHERWISE]:
;    0642  3                             BEGIN
;    0643  4                             Logic_Error ('Loop on Test, Illegal Hook_Point')
;    0644  2                             END;
;    0645  2                    TES;
;    0646  3             RETURN (QA$K_Failure)
;    0647  1     END;


                                                        .PSECT  WORK,NOEXE,2

                                                00000 LOOP_COUNTER:
                                                        .BLKB   2

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 39  002E9 P.AAT:  .ASCII  \9!/**QA Internal Error: Loop on Test, Il\  ;
6F 20 70 6F 6F 4C 20 3A 72 6F 72 72 45 20 6C  002F8                                                           ;
                  6C 49 20 2C 74 73 65 54 20 6E  00307                                                           ;
6E 69 6F 50 5F 6B 6F 6F 48 20 6C 61 67 65 6C  00311          .ASCII  \legal Hook_Point!/\                      ;
                                    2F 21 74  00320                                                           ;

                                                        ES=                 P.AAT


                                                        .PSECT  CODE,NOWRT,  SHR,2

                                          01FC 00000            .ENTRY  QA$LOOP_ON_TEST, Save R2,R3,R4,R5,R6,R7,R8  ; 0518
                                58 00000000G   EF   9E 00002    MOVAB   QA$AOB_CHECK_STATE, R8                      ;
                                57 00000000'   EF   9E 00009    MOVAB   LOOP_COUNTER, R7                            ;
                                56 00000000G   EF   9E 00010    MOVAB   QA$AOB_ROUTINE_STATE, R6                    ;
                                55 00000000G   EF   9E 00017    MOVAB   QA$AOB_FLAGS, R5                            ;
                                54 00000000G   9F   9E 0001E    MOVAB   @#DS$PRINTF, R4                             ;
                                53 00000000'   EF   9E 00025    MOVAB   QA$T_HEADER_1, R3                           ;
                                52         04   AC   D0 0002C    MOVL    HOOK_POINT, R2                              ; 0556
```

```
                                12              52  D1 00030          CMPL    R2, #18                                         ; 0558
                                                39  12 00033          BNEQ    1$                                              ;
                                67              01  B0 00035          MOVW    #1, LOOP_COUNTER                                ; 0568
                                65              10  88 00038          BISB2   #16, QA$AOB_FLAGS                               ; 0569
                                                53  DD 0003B          PUSHL   R3                                              ; 0571
                                64              01  FB 0003D          CALLS   #1, DS$PRINTF                                   ;
                                7E 0000020C     9F  7D 00040          MOVQ    @#^X0000020C, -(SP)                             ; 0572
                                   00000208     9F  DD 00047          PUSHL   @#^X00000208                                    ;
                                         26     A3  9F 0004D          PUSHAB  QA$T_HEADER_2                                   ;
                                64              04  FB 00050          CALLS   #4, DS$PRINTF                                   ;
                                                7E  D4 00053          CLRL    -(SP)                                           ; 0573
                                         4D     A3  9F 00055          PUSHAB  QA$T_HEADER_3                                   ;
                                6/              02  FB 00058          CALLS   #2, DS$PRINTF                                   ;
                                       0087     C3  9F 0005B          PUSHAB  HEADER_LT                                       ; 0574
                                64              01  FB 0005F          CALLS   #1, DS$PRINTF                                   ;
                     00000000G  EF              01  D0 00062          MOVL    #1, DS$GL_CLIBASE+44                            ; 0576
                                66              01  88 00069          BISB2   #1, QA$AOB_ROUTINE_STATE                        ; 0577
                                                58  11 0006C          BRB     7$                                              ; 0578
                                05              52  D1 0006E  1$:      CMPL    R2, #5                                          ; 0581
                                                26  12 00071          BNEQ    4$                                              ;
                             06                 66  02  E1 00073          BBC     #2, QA$AOB_ROUTINE_STATE, 2$                    ; 0583
                                       FEB4     C3  9F 00077          PUSHAB  CLEANUP_ERROR_MESSAGE                           ; 0584
                                                04  11 0007B          BRB     3$                                              ;
                                       FEE4     C3  9F 0007D  2$:      PUSHAB  ERROR_REPORTED_MESSAGE                          ; 0586
                                64              01  FB 00081  3$:      CALLS   #1, DS$PRINTF                                   ;
                                       FF57     C3  9F 00084          PUSHAB  INFINITE_TEST_MESSAGE                           ; 0588
                                64              01  FB 0008E          CALLS   #1, DS$PRINTF                                   ;
                                                02  DD 0008B          PUSHL   #2                                              ; 0590
                     00000000G  EF              01  FB 0008D          CALLS   #1, QA$ADD_QA_ERROR                             ;
                                66              02  88 00094          BISB2   #2, QA$AOB_ROUTINE_STATE                        ; 0591
                                                66  11 00097          BRB     14$                                             ;
                                                52  D5 00099  4$:      TSTL    R2                                              ; 0593
                                                0E  12 0009B          BNEQ    5$                                              ;
                                68              04  8A 0009D          BICB2   #4, QA$AOB_CHECK_STATE                          ; 0599
                                68              08  88 000A0          BISB2   #8, QA$AOB_CHECK_STATE                          ; 0600
                                66              08  88 000A3          BISB2   #8, QA$AOB_ROUTINE_STATE                        ; 0601
                                65              20  88 000A6          BISB2   #32, QA$AOB_FLAGS                               ; 0602
                                                49  11 000A9          BRB     12$                                             ; 0603
                                15              52  D1 000AB  5$:      CMPL    R2, #21                                         ; 0606
                                                18  12 000AE          BNEQ    8$                                              ;
                     00000000G  EF              67  B1 000B0          CMPW    LOOP_COUNTER, QA$W_TESTLOOPS                    ; 0608
                                                08  12 000B7          BNEQ    6$                                              ;
                                67              01  B0 000B9          MOVW    #1, LOOP_COUNTER                                ; 0610
                                65              10  88 000BC          BISB2   #16, QA$AOB_FLAGS                               ; 0611
                                                33  11 000BF          BRB     12$                                             ;
                                                67  B6 000C1  6$:      INCW    LOOP_COUNTER                                    ; 0615
                                65              10  8A 000C3          BICB2   #16, QA$AOB_FLAGS                               ; 0616
                                                2C  11 000C6  7$:      BRB     12$                                             ; 0618
                                                52  D5 000C8  8$:      TSTL    R2                                              ; 0624
                                                05  15 000CA          BLEQ    9$                                              ;
                                02              52  D1 000CC          CMPL    R2, #2                                          ;
                                                23  15 000CF          BLEQ    12$                                             ;
                                06              52  D1 000D1  9$:      CMPL    R2, #6                                          ;
                                                1E  13 000D4          BEQL    12$                                             ;
                                09              52  D1 000D6          CMPL    R2, #9                                          ;
                                                05  19 000D9          BLSS    10$                                             ;
                                0B              52  D1 000DB          CMPL    R2, #11                                         ;
```

```
                                                14  15 000DE             BLEQ    12$                                             ;
                                        11      52  D1 000E0 10$:        CMPL    R2, #17                                         ;
                                                0F  13 000E3             BEQL    12$                                             ;
                                        13      52  D1 000E5             CMPL    R2, #19                                         ;
                                                05  19 000E8             BLSS    11$                                             ;
                                        14      52  D1 000EA             CMPL    R2, #20                                         ;
                                                05  15 000ED             BLEQ    12$                                             ;
                                        18      52  D1 000EF 11$:        CMPL    R2, #24                                         ;
                                                04  12 000F2             BNEQ    13$                                             ;
                                        50      01  D0 000F4 12$:        MOVL    #1, R0                                          ; 0638
                                                    04 000F7             RET                                                     ;
                                            015A    C3  9F 000F8 13$:    PUSHAB  ES                                              ; 0643
                                        64          01  FB 000FC         CALLS   #1, DS$PRINTF                                   ;
                                                    50  D4 000FF 14$:    CLRL    R0                                              ; 0646
                                                        04 00101         RET                                                     ; 0647

; Routine Size:  258 bytes,    Routine Base:  CODE + 019A
```

```
;    0648  1      %SBTTL 'QA$Loop_On_Subtest'
;    0649  1
;    0650  1      GLOBAL ROUTINE QA$Loop_On_Subtest (Hook_Point) =        !                                    Start of [1]
;    0651  1
;    0652  1      !++
;    0653  1      ! FUNCTIONAL DESCRIPTION:
;    0654  1      !
;    0655  1      !       This routine performs the QA Loop on Subtest check.
;    0656  1      !
;    0657  1      ! CALLER(S):
;    0658  1      !
;    0659  1      !       QA$Main
;    0660  1      !
;    0661  1      ! FORMAL PARAMETERS:
;    0662  1      !
;    0663  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0664  1      !                    The constants are defined in the DSQA macro.
;    0665  1      !
;    0666  1      ! IMPLICIT INPUTS:
;    0667  1      !
;    0668  1      !       NONE
;    0669  1      !
;    0670  1      ! IMPLICIT OUTPUTS:
;    0671  1      !
;    0672  1      !       NONE
;    0673  1      !
;    0674  1      ! COMPLETION CODES:
;    0675  1      !
;    0676  1      !       0 - QA error detected; unsuccessful return
;    0677  1      !       1 - No QA errors; successful return
;    0678  1      !
;    0679  1      ! SIDE EFFECTS:
;    0680  1      !
;    0681  1      !       NONE
;    0682  1      !
;    0683  1      !--
```

```
;    0684  2     BEGIN
;    0685  2             MAP                                                 !                                                  [01]
;    0686  2                     DS$GL_CLIBASE : BLOCK [, BYTE];             ! Access the Cli data vector                       [01]
;    0687  2
;    0688  2             SELECTONE .Hook_Point OF
;    0689  2                     SET
;    0690  2                     [DSQA$K_Start_VRReStart]:                   !                                                  [01]
;    0691  3                             BEGIN
;    0692  3                             !+
;    0693  3                             ! The initializations required for this check and that are specific to this check
;    0694  3                             ! must be done here.
;    0695  3                             !+
;    0696  3
;    0697  3                             IF (.QA$AOB_Flags [QA$K_First_Time]) THEN       !                                  [01]
;    0698  4                                     BEGIN
;    0699  4                                     $DS_PRINTF (QA$T_Header_1);
;    0700  4                                     $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0701  4                                     $DS_PRINTF (QA$T_Header_3, 0);
;    0702  4                                     $DS_PRINTF (Header_LS);
;    0703  4
;    0704  4                                     QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done initializing
;    0705  4                                     QA$AOB_Flags [QA$K_First_Time]        = False;  ! No more first time           [01]
;    0706  4                                     QA$AOB_Flags [QA$K_No_Header]         = False;  ! Print the header             [01]
;    0707  4
;    0708  4                                     !+
;    0709  4                                     ! Set up the subtest and pass numbers for looping on a subtest.          [01]
;    0710  4                                     !-
;    0711  4
;    0712  4                                     DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;    ! Set the test to loop on       [01]
;    0713  4                                     DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Test;    ! Set the last test to loop on  [01]
;    0714  4                                     DS$GL_CLIBASE [CLI$L_SUBT] = 1;                  ! Set the subtest to loop on    [01]
;    0715  4                                     DS$GL_CLIBASE [CLI$L_PASS] = .QA$W_SubtestLoops;! Set the no. of loops          [01]
;    0716  4                                     END                             !                                              [01]
```

```
;   0717  3                         ELSE                    !                                                       [01]
;   0718  4                                 BEGIN           !                                                       [01]
;   0719  4                                 !+
;   0720  4                                 ! This is the nth (n > 1) time we have gotten to the VRReStart label. Thus, we  [01]
;   0721  4                                 ! got there by a branch from the EndSub routine, executed when the required     [01]
;   0722  4                                 ! number of loops on a subtest had been done. Thus, go onto the next subtest in [01]
;   0723  4                                 ! the same test. This implies that if the subtest just completed was the last   [01]
;   0724  4                                 ! subtest in that test, then the increment below will cause the subtest to not  [01]
;   0725  4                                 ! be found in the next execution of that test. So, the test will return to the  [01]
;   0726  4                                 ! Dispatch routine, where the test number will be incremented and the subtest   [01]
;   0727  4                                 ! number will be reset to one. Thus, if x is the last subtest number in         [01]
;   0728  4                                 ! test a, and test b follows test a, the sequence will be as follows:           [01]
;   0729  4                                 ! Test a.x-1, VRReStart, Test a.x, VrReStart, Test a, Dispatch, Test b.1,       [01]
;   0730  4                                 ! VrReStart, Test b.2, etc. Note that the above gives only those subtest        [01]
;   0731  4                                 ! numbers (x-1, x, 1, etc.) that will be looped on. Note that all the subtests  [01]
;   0732  4                                 ! prior to that subtest number would be executed before looping on the nth      [01]
;   0733  4                                 ! subtest. I must also reset the Test, Last, and Pass areas in the Cli          [01]
;   0734  4                                 ! data vector because they get destroyed somewhere (Init_Context?).             [01]
;   0735  4                                 !-
;   0736  4
;   0737  4                                 QA$AOB_Flags [QA$K_No_Header] = True;           ! Do not print header anymore   [01]
;   0738  4
;   0739  4                                 DS$GL_CLIBASE [CLI$L_SUBT] = .DS$GL_SUBTEST + 1;! Do next subtest               [01]
;   0740  4                                 DS$GL_CLIBASE [CLI$L_TEST] = .DS$GL_FSTTEST;    ! Reset first test number       [01]
;   0741  4                                 DS$GL_CLIBASE [CLI$L_LAST] = .DS$GL_FSTTEST;    ! Reset last test number        [01]
;   0742  4                                 DS$GL_CLIBASE [CLI$L_PASS] = .QA$W_SubtestLoops;! Reset pass number             [01]
;   0743  3                                 END;                    !                                               [01]
;   0744  4                         RETURN (QA$K_Success)
;   0745  2                         END;
```

```
;    0746  2                    [DSQA$K_Error_Error_End]:
;    0747  3                            BEGIN
;    0748  3                            IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                    [01]
;    0749  4                                    $DS_PRINTF (Cleanup_Error_Message)              !                    [01]
;    0750  3                            ELSE                                                    !                    [01]
;    0751  3                                    $DS_PRINTF (Error_Reported_Message);            !                    [01]
;    0752  3
;    0753  3                            $DS_PRINTF (Infinite_Subtest_Message);                  !                    [02]
;    0754  3
;    0755  3                            QA$Add_QA_Error (QA$K_Loop_On_Subtest);
;    0756  3                            QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0757  2                            END;
;    0758  2
;    0759  2                    [DSQA$K_Dispat_DSX$EndPass_Mid]:
;    0760  3                            BEGIN
;    0761  3                            !+
;    0762  3                            ! All done the Loop on Subtest check. Clean up. Restore the first and last test    [01]
;    0763  3                            ! numbers just in case....                                                        [01]
;    0764  3                            !-
;    0765  3
;    0766  3                            DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! Restore the old first test #   [01]
;    0767  3                            DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last;   ! Restore the old last test #    [01]
;    0768  3
;    0769  3                            QA$AOB_Check_State [QA$K_Loop_On_Subtest] = Off;
;    0770  3                            QA$AOB_Check_State [QA$K_Run_Backwards]   = On;
;    0771  3                            QA$AOB_Routine_State [QA$V_Check_Done]    = True;
;    0772  3                            QA$AOB_Flags [QA$K_No_Header]             = False;      ! Print header again  [01]
;    0773  4                            RETURN (QA$K_Success)
;    0774  2                            END;
```

```
;    0775  2                    !+
;    0776  2                    ! The following are not used in this check.
;    0777  2                    !-
;    0778  2
;    0779  2                    [DSQA$K_Dispat_Before_Init,
;    0780  2                     DSQA$K_Dispat_After_Init,                !                                  [01]
;    0781  2                     DSQA$K_Dispat_CallTest,                  !                                  [01]
;    0782  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,           !                                  [01]
;    0783  2                     DSQA$K_Dispat_DSX$EndPass_End,           !                                  [01]
;    0784  2                     DSQA$K_Dispat_Done_Tests,                !                                  [01]
;    0785  2
;    0786  2                     DSQA$K_Error_Error_Bgn,                  !                                  [01]
;    0787  2
;    0788  2                     DSQA$K_Loop_DSX$BgnSub,
;    0789  2                     DSQA$K_Loop_DSX$EndSub,
;    0790  2                     DSQA$K_Loop_DSX$CkLoop,
;    0791  2
;    0792  2                     DSQA$K_QA_DSX$Branch]:
;    0793  3                             BEGIN
;    0794  4                             RETURN (QA$K_Success)
;    0795  2                             END;
;    0796  2
;    0797  2                    [OTHERWISE]:
;    0798  3                             BEGIN
;    0799  4                             Logic_Error ('Loop on Subtest, Illegal Hook_Point')
;    0800  2                             END;
;    0801  2                    TES;
;    0802  3             RETURN (QA$K_Failure)
;    0803  1     END;


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 3C  00323 P.AAU:  .ASCII  \<!/**QA Internal Error: Loop on Subtest,\  ;
6F 20 70 6F 6F 4C 20 3A 72 6F 72 72 45 20 6C  00332                                                          ;
                  2C 74 73 65 74 62 75 53 20 6E  00341                                                          ;
50 5F 6B 6F 6F 48 20 6C 61 67 65 6C 6C 49 20  00348          .ASCII  \ Illegal Hook_Point!/\                 ;
                     2F 21 74 6E 69 6F        0035A                                                          ;

                                                ES=                 P.AAU


                                                        .PSECT  CODE,NOWRT,  SHR,2

                                          07FC 00000    .ENTRY  QA$LOOP_ON_SUBTEST, Save R2,R3,R4,R5,R6,R7,-; 0650
                                                                R8,R9,R10                                       ;
                                5A 00000000G   EF  9E 00002  MOVAB   QA$AOB_CHECK_STATE, R10                  ;
                                59 00000000G   EF  9E 00009  MOVAB   DS$GL_FSTTEST, R9                        ;
                                58 00000000G   EF  9E 00010  MOVAB   QA$W_SAVE_TEST, R8                       ;
                                57 00000000G   EF  9E 00017  MOVAB   QA$AOB_FLAGS, R7                         ;
                                56 00000000G   EF  9E 0001E  MOVAB   QA$AOB_ROUTINE_STATE, R6                 ;
                                55 00000000G   9F  9E 00025  MOVAB   @#DS$PRINTF, R5                          ;
                                54 00000000'   EF  9E 0002C  MOVAB   QA$T_HEADER_1, R4                        ;
                                53 00000000G   EF  9E 00033  MOVAB   DS$GL_CLIBASE+32, R3                     ;
                                52       04    AC  D0 0003A  MOVL    HOOK_POINT, R2                          ; 0688
                                12             52  D1 0003E  CMPL    R2, #18                                 ; 0690
```

```
                                           5D  12 00041          BNEQ    3$                                                  ;
               3B              67          05  E1 00043          BBC     #5, QA$AOB_FLAGS, 1$                                ; 0697
                                           54  DD 00047          PUSHL   R4                                                  ; 0699
                               65          01  FB 00049          CALLS   #1, DS$PRINTF                                       ;
                               7E 0000020C 9F  7D 0004C          MOVQ    @#^X0000020C, -(SP)                                 ; 0700
                                  00000208 9F  DD 00053          PUSHL   @#^X00000208                                        ;
                                        26 A4  9F 00059          PUSHAB  QA$T_HEADER_2                                       ;
                               65          04  FB 0005C          CALLS   #4, DS$PRINTF                                       ;
                                           7E  D4 0005F          CLRL    -(SP)                                               ; 0701
                                        4D A4  9F 00061          PUSHAB  QA$T_HEADER_3                                       ;
                               65          02  FB 00064          CALLS   #2, DS$PRINTF                                       ;
                                      00A6 C4  9F 00067          PUSHAB  HEADER_LS                                           ; 0702
                               65          01  FB 0006B          CALLS   #1, DS$PRINTF                                       ;
                               66          01  88 0006E          BISB2   #1, QA$AOB_ROUTINE_STATE                            ; 0704
                               67       60 8F  8A 00071          BICB2   #96, QA$AOB_FLAGS                                   ; 0706
                               63          68  32 00075          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+32                    ; 0712
                          04   A3          68  32 00078          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+36                    ; 0713
                          08   A3          01  D0 0007C          MOVL    #1, DS$GL_CLIBASE+40                                ; 0714
                                           14  11 00080          BRB     2$                                                  ; 0697
                               67       40 8F  88 00082 1$:      BISB2   #64, QA$AOB_FLAGS                                   ; 0737
          08   A3 00000000G    EF          01  C1 00086          ADDL3   #1, DS$GL_SUBTEST, DS$GL_CLIBASE+40                 ; 0739
                               63          69  D0 0008F          MOVL    DS$GL_FSTTEST, DS$GL_CLIBASE+32                     ; 0740
                          04   A3          69  D0 00092          MOVL    DS$GL_FSTTEST, DS$GL_CLIBASE+36                     ; 0741
                          0C   A3 00000000G EF 3C 00096 2$:      MOVZWL  QA$W_SUBTESTLOOPS, DS$GL_CLIBASE+44                 ; 0715
                                           72  11 0009E          BRB     11$                                                 ; 0744
                               05          52  D1 000A0 3$:      CMPL    R2, #5                                              ; 0746
                                           25  12 000A3          BNEQ    6$                                                  ;
               06              66          02  E1 000A5          BBC     #2, QA$AOB_ROUTINE_STATE, 4$                        ; 0748
                                      FEB4 C4  9F 000A9          PUSHAB  CLEANUP_ERROR_MESSAGE                               ; 0749
                                           04  11 000AD          BRB     5$                                                  ;
                                      FEE4 C4  9F 000AF 4$:      PUSHAB  ERROR_REPORTED_MESSAGE                              ; 0751
                               65          01  FB 000B3 5$:      CALLS   #1, DS$PRINTF                                       ;
                                        89 A4  9F 000B6          PUSHAB  INFINITE_SUBTEST_MESSAGE                            ; 0753
                               65          01  FB 000B9          CALLS   #1, DS$PRINTF                                       ;
                                           03  DD 000BC          PUSHL   #3                                                  ; 0755
                  00000000G    EF          01  FB 000BE          CALLS   #1, QA$ADD_QA_ERROR                                 ;
                               66          02  88 000C5          BISB2   #2, QA$AOB_ROUTINE_STATE                            ; 0756
                                           53  11 000C8          BRB     13$                                                 ;
                                           52  D5 000CA 6$:      TSTL    R2                                                  ; 0759
                                           1A  12 000CC          BNEQ    7$                                                  ;
                               63          68  32 000CE          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+32                    ; 0766
                          04   A3 00000000G EF 32 000D1          CVTWL   QA$W_SAVE_LAST, DS$GL_CLIBASE+36                    ; 0767
                               6A          08  8A 000D9          BICB2   #8, QA$AOB_CHECK_STATE                              ; 0769
                               6A          10  88 000DC          BISB2   #16, QA$AOB_CHECK_STATE                             ; 0770
                               66          08  88 000DF          BISB2   #8, QA$AOB_ROUTINE_STATE                            ; 0771
                               67       40 8F  8A 000E2          BICB2   #64, QA$AOB_FLAGS                                   ; 0772
                                           2A  11 000E6          BRB     11$                                                 ; 0773
                                           05  15 000E8 7$:      BLEQ    8$                                                  ; 0779
                               02          52  D1 000EA          CMPL    R2, #2                                              ;
                                           23  15 000ED          BLEQ    11$                                                 ;
                               06          52  D1 000EF 8$:      CMPL    R2, #6                                              ;
                                           1E  13 000F2          BEQL    11$                                                 ;
                               09          52  D1 000F4          CMPL    R2, #9                                              ;
                                           05  19 000F7          BLSS    9$                                                  ;
                               0B          52  D1 000F9          CMPL    R2, #11                                             ;
                                           14  15 000FC          BLEQ    11$                                                 ;
                               11          52  D1 000FE 9$:      CMPL    R2, #17                                             ;
```

```
                                 0F  13 00101          BEQL    11$                                     ;
                          13     52  D1 00103          CMPL    R2, #19                                 ;
                                 05  19 00106          BLSS    10$                                     ;
                          15     52  D1 00108          CMPL    R2, #21                                 ;
                                 05  15 0010B          BLEQ    11$                                     ;
                          18     52  D1 0010D 10$:      CMPL    R2, #24                                 ;
                                 04  12 00110          BNEQ    12$                                     ;
                          50     01  D0 00112 11$:      MOVL    #1, R0                                  ; 0794
                                 04  00115            RET                                             ;
                    0194  C4  9F  00116 12$:      PUSHAB  ES                                      ; 0799
                          65     01  FB 0011A          CALLS   #1, DS$PRINTF                           ;
                                 50  D4 0011D 13$:      CLRL    R0                                      ; 0802
                                     04 0011F          RET                                             ; 0803

; Routine Size:  288 bytes,    Routine Base:  CODE + 029C
```

```
;    0804  1     %SBTTL 'QA$Run_Backwards'
;    0805  1
;    0806  1     GLOBAL ROUTINE QA$Run_Backwards (Hook_Point) =
;    0807  1
;    0808  1     !++
;    0809  1     ! FUNCTIONAL DESCRIPTION:
;    0810  1     !
;    0811  1     !       This routine performs the QA Run Backwards check.
;    0812  1     !
;    0813  1     ! CALLER(S):
;    0814  1     !
;    0815  1     !       QA$Main
;    0816  1     !
;    0817  1     ! FORMAL PARAMETERS:
;    0818  1     !
;    0819  1     !       Hook_Point : The point in the Supervisor from which this routine
;    0820  1     !                    was called. This is a constant value. The constants are
;    0821  1     !                    defined in the DSQA macro.
;    0822  1     !
;    0823  1     ! IMPLICIT INPUTS:
;    0824  1     !
;    0825  1     !       NONE
;    0826  1     !
;    0827  1     ! IMPLICIT OUTPUTS:
;    0828  1     !
;    0829  1     !       NONE
;    0830  1     !
;    0831  1     ! COMPLETION CODES:
;    0832  1     !
;    0833  1     !       0 - QA error detected; unsuccessful return
;    0834  1     !       1 - No QA errors; successful return
;    0835  1     !
;    0836  1     ! SIDE EFFECTS:
;    0837  1     !
;    0838  1     !       NONE
;    0839  1     !
;    0840  1     !--
```

```
;    0841  2     BEGIN
;    0842  2             SELECTONE .Hook_Point OF
;    0843  2                     SET
;    0844  2                     [DSQA$K_Start_VRReStart]:               !                                                [01]
;    0845  3                             BEGIN
;    0846  3                             !+
;    0847  3                             ! The initializations required for this check and that are specific to this check
;    0848  3                             ! must be done here.
;    0849  3                             !+
;    0850  3
;    0851  3                             $DS_PRINTF (QA$T_Header_1);
;    0852  3                             $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0853  3                             $DS_PRINTF (QA$T_Header_3, 0);
;    0854  3                             $DS_PRINTF (Header_RB);
;    0855  3
;    0856  3                             QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0857  4                             RETURN (QA$K_Success)
;    0858  2                             END;
;    0859  2
;    0860  2                     [DSQA$K_Error_Error_End]:
;    0861  3                             BEGIN
;    0862  3                             IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                        [01]
;    0863  4                                     $DS_PRINTF (Cleanup_Error_Message)              !                        [01]
;    0864  3                             ELSE                                                    !                        [01]
;    0865  3                                     $DS_PRINTF (Error_Reported_Message);            !                        [01]
;    0866  3
;    0867  3                             $DS_PRINTF (Run_Backwards_Message);             !                                [02]
;    0868  3
;    0869  3                             QA$Add_QA_Error (QA$K_Run_Backwards);
;    0870  3                             QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0871  2                             END;
;    0872  2
;    0873  2                     [DSQA$K_Dispat_DSX$EndPass_Mid]:        !                                                [01]
;    0874  3                             BEGIN
;    0875  3                             QA$AOB_Check_State [QA$K_Run_Backwards]   = Off;
;    0876  3                             QA$AOB_Routine_State [QA$V_Check_Done]    = True;
;    0877  3                             QA$AOB_Flags [QA$K_First_Time]            = True;       ! For the next check   [01]
;    0878  4                             RETURN (QA$K_Success)
;    0879  2                             END;
;    0880  2
;    0881  2                     [DSQA$K_Dispat_After_Init]:             !                                                [01]
;    0882  3                             BEGIN                          !                                                [01]
;    0883  3                             DSA$GL_TESTNO = .DS$GL_LSTTEST; ! Start at the last requested test number (i.e.,  [01]
;    0884  3                                                             ! at 'y' in /TEST:x:y), or else at L$A_TSTCNT.   [01]
;    0885  4                             RETURN (QA$K_Success)          !                                                [01]
;    0886  2                             END;
```

```
;    0887  2                    !+
;    0888  2                    ! The following are not used in this check.
;    0889  2                    !-
;    0890  2
;    0891  2                    [DSQA$K_Dispat_Before_Init,
;    0892  2                     DSQA$K_Dispat_CallTest,                 !                                [01]
;    0893  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,          !                                [01]
;    0894  2                     DSQA$K_Dispat_DSX$EndPass_End,          !                                [01]
;    0895  2                     DSQA$K_Dispat_Done_Tests,               !                                [01]
;    0896  2
;    0897  2                     DSQA$K_Error_Error_Bgn,                 !                                [01]
;    0898  2
;    0899  2                     DSQA$K_Loop_DSX$BgnSub,
;    0900  2                     DSQA$K_Loop_DSX$EndSub,
;    0901  2                     DSQA$K_Loop_DSX$CkLoop,
;    0902  2
;    0903  2                     DSQA$K_QA_DSX$Branch]:
;    0904  3                             BEGIN
;    0905  4                             RETURN (QA$K_Success)
;    0906  2                             END;
;    0907  2
;    0908  2                    [OTHERWISE]:
;    0909  3                             BEGIN
;    0910  4                             Logic_Error ('Run Backwards, Illegal Hook_Point')
;    0911  2                             END;
;    0912  2                    TES;
;    0913  3             RETURN (QA$K_Failure)
;    0914  1     END;


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 3A 00360 P.AAV:  .ASCII  \:!/**QA Internal Error: Run Backwards, I\  ;
61 42 20 6E 75 52 20 3A 72 6F 72 72 45 20 6C 0036F                                                        ;
                  49 20 2C 73 64 72 61 77 6B 63 0037E                                                     ;
69 6F 50 5F 6B 6F 6F 48 20 6C 61 67 65 6C 6C 00388        .ASCII  \llegal Hook_Point!/\                   ;
                              2F 21 74 6E 00397                                                           ;

                                                  ES=                   P.AAV

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                  003C 00000            .ENTRY  QA$RUN_BACKWARDS, Save R2,R3,R4,R5      ; 0806
                                55 00000000G    EF   9E 00002        MOVAB   QA$AOB_ROUTINE_STATE, R5                ;
                                54 00000000G    9F   9E 00009        MOVAB   @#DS$PRINTF, R4                         ;
                                53 00000000'    EF   9E 00010        MOVAB   QA$T_HEADER_1, R3                       ;
                                52        04    AC   D0 00017        MOVL    HOOK_POINT, R2                          ; 0842
                                12              52   D1 0001B        CMPL    R2, #18                                 ; 0844
                                                2C   12 0001E        BNEQ    1$                                      ;
                                                53   DD 00020        PUSHL   R3                                      ; 0851
                                64              01   FB 00022        CALLS   #1, DS$PRINTF                           ;
                                7E 0000020C     9F   7D 00025        MOVQ    @#^X0000020C, -(SP)                     ; 0852
                                   00000208     9F   DD 0002C        PUSHL   @#^X00000208                            ;
                                         26     A3   9F 00032        PUSHAB  QA$T_HEADER_2                           ;
                                64              04   FB 00035        CALLS   #4, DS$PRINTF                           ;
```

```
                                                7E  D4 00038          CLRL    -(SP)                                          ; 0853
                                             4D A3  9F 0003A          PUSHAB  QA$T_HEADER_3                                  ;
                                          64    02  FB 0003D          CALLS   #2, DS$PRINTF                                  ;
                                        00C8    C3  9F 00040          PUSHAB  HEADER_RB                                      ; 0854
                                          64    01  FB 00044          CALLS   #1, DS$PRINTF                                  ;
                                          65    01  88 00047          BISB2   #1, QA$AOB_ROUTINE_STATE                       ; 0856
                                                7F  11 0004A          BRB     9$                                             ; 0857
                                          05    52  D1 0004C 1$:      CMPL    R2, #5                                         ; 0860
                                                25  12 0004F          BNEQ    4$                                             ;
                                          65    02  E1 00051          BBC     #2, QA$AOB_ROUTINE_STATE, 2$                   ; 0862
                                        FEB4    C3  9F 00055          PUSHAB  CLEANUP_ERROR_MESSAGE                          ; 0863
                                                04  11 00059          BRB     3$                                             ;
                                        FEE4    C3  9F 0005B 2$:      PUSHAB  ERROR_REPORTED_MESSAGE                         ; 0865
                                          64    01  FB 0005F 3$:      CALLS   #1, DS$PRINTF                                  ;
                                          BE    A3  9F 00062          PUSHAB  RUN_BACKWARDS_MESSAGE                          ; 0867
                                          64    01  FB 00065          CALLS   #1, DS$PRINTF                                  ;
                                                04  DD 00068          PUSHL   #4                                             ; 0869
                                00000000G   EF      01  FB 0006A      CALLS   #1, QA$ADD_QA_ERROR                            ;
                                          65    02  88 00071          BISB2   #2, QA$AOB_ROUTINE_STATE                       ; 0870
                                                60  11 00074          BRB     11$                                            ;
                                                52  D5 00076 4$:      TSTL    R2                                             ; 0873
                                                13  12 00078          BNEQ    5$                                             ;
                                00000000G   EF      10  8A 0007A      BICB2   #16, QA$AOB_CHECK_STATE                        ; 0875
                                          65    08  88 00081          BISB2   #8, QA$AOB_ROUTINE_STATE                       ; 0876
                                00000000G   EF      20  88 00084      BISB2   #32, QA$AOB_FLAGS                              ; 0877
                                                3E  11 0008B          BRB     9$                                             ; 0878
                                          14    52  D1 0008D 5$:      CMPL    R2, #20                                        ; 0881
                                                0D  12 00090          BNEQ    6$                                             ;
                                0000FE50  9F 00000000G  EF  D0 00092  MOVL    DS$GL_LSTTEST, @#^X0000FE50                    ; 0883
                                                2C  11 0009D          BRB     9$                                             ; 0885
                                                52  D5 0009F 6$:      TSTL    R2                                             ; 0891
                                                05  15 000A1          BLEQ    7$                                             ;
                                          02    52  D1 000A3          CMPL    R2, #2                                         ;
                                                23  15 000A6          BLEQ    9$                                             ;
                                          06    52  D1 000A8 7$:      CMPL    R2, #6                                         ;
                                                1E  13 000AB          BEQL    9$                                             ;
                                          09    52  D1 000AD          CMPL    R2, #9                                         ;
                                                05  19 000B0          BLSS    8$                                             ;
                                          08    52  D1 000B2          CMPL    R2, #11                                        ;
                                                14  15 000B5          BLEQ    9$                                             ;
                                          11    52  D1 000B7 8$:      CMPL    R2, #17                                        ;
                                                0F  13 000BA          BEQL    9$                                             ;
                                          13    52  D1 000BC          CMPL    R2, #19                                        ;
                                                0A  13 000BF          BEQL    9$                                             ;
                                          15    52  D1 000C1          CMPL    R2, #21                                        ;
                                                05  13 000C4          BEQL    9$                                             ;
                                          18    52  D1 000C6          CMPL    R2, #24                                        ;
                                                04  12 000C9          BNEQ    10$                                            ;
                                          50    01  D0 000CB 9$:      MOVL    #1, R0                                         ; 0905
                                                04  000CE            RET                                                    ;
                                        01D1    C3  9F 000CF 10$:     PUSHAB  ES                                             ; 0910
                                          64    01  FB 000D3          CALLS   #1, DS$PRINTF                                  ;
                                                50  D4 000D6 11$:     CLRL    R0                                             ; 0913
                                                    04 000D8          RET                                                    ; 0914

; Routine Size:  217 bytes,    Routine Base:  CODE + 038C
```

```
;    0915  1      !%SBTTL 'QA$Error_Phase_One'
;    0916  1      !
;    0917  1      !GLOBAL ROUTINE QA$Error_Phase_One (Hook_Point) =       !                                   Start of [1]
;    0918  1      !
;    0919  1      !++
;    0920  1      ! FUNCTIONAL DESCRIPTION:
;    0921  1      !
;    0922  1      !     This routine performs Phase One of the QA Error check.
;    0923  1      !
;    0924  1      ! CALLER(S):
;    0925  1      !
;    0926  1      !     QA$Main
;    0927  1      !
;    0928  1      ! FORMAL PARAMETERS:
;    0929  1      !
;    0930  1      !     Hook_Point : The point in the Supervisor from which this routine
;    0931  1      !                  was called. This is a constant value. The constants are
;    0932  1      !                  defined in the DSQA macro.
;    0933  1      !
;    0934  1      ! IMPLICIT INPUTS:
;    0935  1      !
;    0936  1      !     NONE
;    0937  1      !
;    0938  1      ! IMPLICIT OUTPUTS:
;    0939  1      !
;    0940  1      !     NONE
;    0941  1      !
;    0942  1      ! COMPLETION CODES:
;    0943  1      !
;    0944  1      !     0 - QA error detected; unsuccessful return
;    0945  1      !     1 - No QA errors; successful return
;    0946  1      !
;    0947  1      ! SIDE EFFECTS:
;    0948  1      !
;    0949  1      !--
```

```
;    0950  1      !
;    0951  1      !BEGIN
;    0952  1      !       OWN
;    0953  1      !               Abort_Now : BYTE,                       ! When 1, abort the diagnostic
;    0954  1      !               Last_Branch_PC;                         ! The PC of the last branch macro executed
;    0955  1      !
;    0956  1      !       MAP
;    0957  1      !               DS$GL_CLIBASE : BLOCK [, BYTE];
;    0958  1      !
;    0959  1      !       LITERAL                                         ! Used to set the Last_Branch_PC so as to insure
;    0960  1      !               QA$K_Bad_PC = %X'FFFFFFFF';             ! that the PC is not found.
;    0961  1      !
;    0962  1      !       SELECTONE .Hook_Point OF
;    0963  1      !               SET
;    0964  1      !               [DSQA$K_Start_VRReStart]:
;    0965  1      !                       BEGIN
;    0966  1      !                       IF (.QA$AOB_Flags [QA$K_First_Time]) THEN
;    0967  1      !                               BEGIN
;    0968  1      !                               $DS_PRINTF (QA$T_Header_1);
;    0969  1      !                               $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0970  1      !                               $DS_PRINTF (QA$T_Header_3, 0);                  ! 0 is for the current time
;    0971  1      !                               $DS_PRINTF (Header_E1);
;    0972  1      !
;    0973  1      !                               Last_Branch_PC = 0;                             ! Zero means do next error call
;    0974  1      !                               Abort_Now = False;                              ! Do not abort on next one
;    0975  1      !                               QA$AOB_Flags [QA$K_First_Time] = False;         ! Not first time anymore
;    0976  1      !                               QA$AOB_Flags [QA$K_No_Header]  = False;         ! Print the header only once
;    0977  1      !
;    0978  1      !                               DS$GL_CLIBASE [CLI$L_PASS] = 1;                 ! One pass
;    0979  1      !                               DS$GL_CLIBASE [CLI$L_SUBT] = 0;                 ! No looping on subtests
;    0980  1      !                               DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! First test number
;    0981  1      !                               DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last;   ! Last test number
;    0982  1      !
;    0983  1      !                               QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done initializing
;    0984  1      !                               END
;    0985  1      !                       ELSE
;    0986  1      !                               BEGIN
;    0987  1      !                               QA$AOB_Flags [QA$K_No_Header] = True;           ! Print no more header till done
;    0988  1      !
;    0989  1      !                               DS$GL_CLIBASE [CLI$L_PASS] = 1;                 ! One pass
;    0990  1      !                               DS$GL_CLIBASE [CLI$L_SUBT] = 0;                 ! No looping on subtests
;    0991  1      !                               DS$GL_CLIBASE [CLI$L_TEST] = .DSA$GL_TestNo;    ! Set to same test number
;    0992  1      !                               DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last;   ! Last test number
;    0993  1      !                               END;
;    0994  1      !                       RETURN (QA$K_Success)
;    0995  1      !                       END;
```

```
;    0996  1        !
;    0997  1        !               [DSQA$K_Error_Error_End]:
;    0998  1        !                       BEGIN
;    0999  1        !                       IF ((.Abort_Now) AND (NOT .QA$AOB_Routine_State [QA$V_Doing_Cleanup])) THEN
;    1000  1        !                               BEGIN
;    1001  1        !                               MAP
;    1002  1        !                                       DSA$GL_ERRNO : WORD;                    ! In the DS, it is a word.
;    1003  1        !                               !+
;    1004  1        !                               ! This error call was forced. Add the error number to the forced error summary table
;    1005  1        !                               ! and abort the diagnostic. This will cause the DSX$Abort routine to branch back to
;    1006  1        !                               ! the VRReStart routine, and start over again. Note: use the error number that the
;    1007  1        !                               ! RRepError routine stores in DSA$GL_ErrNo, rather than that passes to the Branch
;    1008  1        !                               ! routine. The former number is more accurate.
;    1009  1        !                               !-
;    1010  1        !                               QA$Add_Forced_Error (.DSA$GL_ERRNO);            ! Add another forced error
;    1011  1        !                               Abort_Now = False;                              ! Reset for next time
;    1012  1        !                               DSX$Abort ();                                   ! Abort the diagnostic
;    1013  1        !                               !+
;    1014  1        !                               ! The above Abort will not come back to here.
;    1015  1        !                               !-
;    1016  1        !                               END
;    1017  1        !                       ELSE
;    1018  1        !                               BEGIN
;    1019  1        !                               !+
;    1020  1        !                               ! This error call was not forced. Therefore, it is a QA error. Print the
;    1021  1        !                               ! appropriate error message and return Failure.
;    1022  1        !                               !-
;    1023  1        !                               IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN
;    1024  1        !                                       $DS_PRINTF (Cleanup_Error_Message)
;    1025  1        !                               ELSE
;    1026  1        !                                       $DS_PRINTF (Error_Reported_Message);
;    1027  1        !                               QA$Add_QA_Error (QA$K_Error_Phase_One);
;    1028  1        !                               QA$AOB_Routine_State [QA$V_Error_Found] = True;
;    1029  1        !                               END;
;    1030  1        !                       !+
;    1031  1        !                       ! Return Failure.
;    1032  1        !                       !-
;    1033  1        !                       END;
;    1034  1        !
;    1035  1        !               [DSQA$K_Dispat_Done_Tests]:
;    1036  1        !                       BEGIN
;    1037  1        !                       !+
;    1038  1        !                       ! We have done all the tests. Set Abort_Now to False so that the error macros in
;    1039  1        !                       ! the initialization code are not done again.
;    1040  1        !                       !-
;    1041  1        !                       Abort_Now      = False;
;    1042  1        !                       Last_Branch_PC = QA$K_Bad_PC;
;    1043  1        !                       RETURN (QA$K_Success)
;    1044  1        !                       END;
```

```
;    1045  1     !
;    1046  1     !                  [DSQA$K_Dispat_DSX$EndPass_Mid]:
;    1047  1     !                          BEGIN
;    1048  1     !                          QA$AOB_Check_State [QA$K_Error_Phase_One]  = Off;       ! Stop Phase One check.
;    1049  1     !                          QA$AOB_Check_State [QA$K_Error_Phase_Two]  = On;        ! Start Phase Two check.
;    1050  1     !                          QA$ACB_Flags [QA$K_No_Header]              = False;     ! Print headers again.
;    1051  1     !
;    1052  1     !                          QA$Print_Forced_Errors ();                              ! Print the table.
;    1053  1     !
;    1054  1     !                          QA$AOB_Routine_State [QA$V_Check_Done]     = True;      ! Indicate end of this check.
;    1055  1     !                          RETURN (QA$K_Success)
;    1056  1     !                          END;
;    1057  1     !
;    1058  1     !                  [DSQA$K_QA_DSX$Branch]:
;    1059  1     !                          BEGIN
;    1060  1     !                          LOCAL
;    1061  1     !                                  Frame_Start,
;    1062  1     !                                  Saved_PC;
;    1063  1     !
;    1064  1     !                          IF (NOT QA$Find_User_Call_Frame (Frame_Start)) THEN
;    1065  1     !                                  RETURN (QA$K_Failure);
;    1066  1     !
;    1067  1     !                          BEGIN
;    1068  1     !                                  BIND
;    1069  1     !                                          Call_Frame = .Frame_Start : BLOCK [, BYTE];
;    1070  1     !
;    1071  1     !                                  Saved_PC = .Call_Frame [SF$L_SAVE_PC]
;    1072  1     !                          END;
;    1073  1     !
;    1074  1     !                          IF (.Last_Branch_PC EQL 0) THEN
;    1075  1     !                                  BEGIN
;    1076  1     !                                  Last_Branch_PC = .Saved_PC;
;    1077  1     !                                  Abort_Now      = True;
;    1078  1     !                                  QA$W_Branch    = Boolean (NOT .QA$W_Branch);
;    1079  1     !                                  END
;    1080  1     !                          ELSE IF (.Last_Branch_PC EQL .Saved_PC) THEN
;    1081  1     !                                  BEGIN
;    1082  1     !                                  Last_Branch_PC = 0;
;    1083  1     !                                  END;
;    1084  1     !                          !+
;    1085  1     !                          ! ELSE:
;    1086  1     !                          !       I have not yet found the correct branch macro. Keep looking.
;    1087  1     !                          !-
;    1088  1     !                          RETURN (QA$K_Success)
;    1089  1     !                          END;
```

```
;    1090  1      !
;    1091  1      !               !+
;    1092  1      !               ! The following are not used in this check. So, just return Success for them.
;    1093  1      !               !-
;    1094  1      !
;    1095  1      !               [DSQA$K_Dispat_Before_Init,
;    1096  1      !                DSQA$K_Dispat_After_Init,
;    1097  1      !                DSQA$K_Dispat_CallTest,
;    1098  1      !                DSQA$K_Dispat_DSX$EndPass_Bgn,          !                                    [01]
;    1099  1      !                DSQA$K_Dispat_DSX$EndPass_End,          !                                    [01]
;    1100  1      !
;    1101  1      !                DSQA$K_Error_Error_Bgn,                 !                                    [01]
;    1102  1      !
;    1103  1      !                DSQA$K_Loop_DSX$BgnSub,
;    1104  1      !                DSQA$K_Loop_DSX$EndSub,
;    1105  1      !                DSQA$K_Loop_DSX$CkLoop]:
;    1106  1      !                        BEGIN
;    1107  1      !                        RETURN (QA$K_Success)
;    1108  1      !                        END;
;    1109  1      !
;    1110  1      !               [OTHERWISE]:
;    1111  1      !                        BEGIN
;    1112  1      !                        Logic_Error ('Error Phase One, Illegal Hook_Point')
;    1113  1      !                        END;
;    1114  1      !               TES;
;    1115  1      !
;    1116  1      !       !+
;    1117  1      !       ! Come to here only if a QA error was found. Return QA$K_Failure to QA$Main, which then takes
;    1118  1      !       ! the appropriate action.
;    1119  1      !       !-
;    1120  1      !
;    1121  1      !       RETURN (QA$K_Failure)
;    1122  1      !END;
```

```
;    1123  1      !
;    1124  1      !%SBTTL 'QA$Error_Phase_Two'
;    1125  1      !
;    1126  1      !GLOBAL ROUTINE QA$Error_Phase_Two (Hook_Point) =
;    1127  1      !
;    1128  1      !++
;    1129  1      ! FUNCTIONAL DESCRIPTION:
;    1130  1      !
;    1131  1      !     This routine performs Phase Two of the QA Error check.
;    1132  1      !
;    1133  1      ! CALLER(S):
;    1134  1      !
;    1135  1      !     QA$Main
;    1136  1      !
;    1137  1      ! FORMAL PARAMETERS:
;    1138  1      !
;    1139  1      !     Hook_Point : The point in the Supervisor from which this routine
;    1140  1      !                  was called. This is a constant value. The constants are
;    1141  1      !                  defined in the DSQA macro.
;    1142  1      !
;    1143  1      ! IMPLICIT INPUTS:
;    1144  1      !
;    1145  1      !     NONE
;    1146  1      !
;    1147  1      ! IMPLICIT OUTPUTS:
;    1148  1      !
;    1149  1      !     NONE
;    1150  1      !
;    1151  1      ! COMPLETION CODES:
;    1152  1      !
;    1153  1      !     0 - QA error detected; unsuccessful return
;    1154  1      !     1 - No QA errors; successful return
;    1155  1      !
;    1156  1      ! SIDE EFFECTS:
;    1157  1      !
;    1158  1      !--
;    1159  1      !
;    1160  1      !BEGIN
;    1161  1      !     RETURN (QA$K_Failure)
;    1162  1      !END;
```

```
;    1163  1      !
;    1164  1      %SBTTL 'QA$Error_Phase_Three'
;    1165  1      !
;    1166  1      GLOBAL ROUTINE QA$Error_Phase_Three (Hook_Point) =
;    1167  1      !
;    1168  1      !++
;    1169  1      ! FUNCTIONAL DESCRIPTION:
;    1170  1      !
;    1171  1      !       This routine performs Phase Three of the QA Error check.
;    1172  1      !
;    1173  1      ! CALLER(S):
;    1174  1      !
;    1175  1      !       QA$Main
;    1176  1      !
;    1177  1      ! FORMAL PARAMETERS:
;    1178  1      !
;    1179  1      !       Hook_Point : The point in the Supervisor from which this routine
;    1180  1      !                    was called. This is a constant value. The constants are
;    1181  1      !                    defined in the DSQA macro.
;    1182  1      !
;    1183  1      ! IMPLICIT INPUTS:
;    1184  1      !
;    1185  1      !       NONE
;    1186  1      !
;    1187  1      ! IMPLICIT OUTPUTS:
;    1188  1      !
;    1189  1      !       NONE
;    1190  1      !
;    1191  1      ! COMPLETION CODES:
;    1192  1      !
;    1193  1      !       0 - QA error detected; unsuccessful return
;    1194  1      !       1 - No QA errors; successful return
;    1195  1      !
;    1196  1      ! SIDE EFFECTS:
;    1197  1      !
;    1198  1      !--
;    1199  1      !
;    1200  1      BEGIN
;    1201  1              RETURN (QA$K_Failure)
;    1202  1      END;
```

```
;   1203  1     %SBTTL 'End of Module QACHECKS'
;   1204  1
;   1205  0     END ELUDOM
```

```
;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  DATA                           923  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          1173  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                             2  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                               Library Statistics

;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time

;  DRB1:[DS.WORK]DIAG.L32;265              784         8         1          85        00:00.2
;  DRB1:[DS.WORK]DS.L32;159                653         7         1          42        00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7         18017         3         0         975        00:04.8
```

```
;                                    COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE QACHECKS

; Size:          1173 code + 925 data bytes
; Run Time:         00:30.0
; Elapsed Time:     01:56.7
; Lines/CPU Min:     2411
; Lexemes/CPU-Min: 22040
; Memory Used:  152 pages
; Compilation Complete
```

```
;    0001  0      %TITLE '*** Setting and Showing QA Flags'
;    0002  0      MODULE QAFLAGS (
;    0003  0                IDENT = '6.6-04'
;    0004  0                ) =
;    0005  0      !
;    0006  0      ! COPYRIGHT (c) 1979, 1981, 1983
;    0007  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0008  0      !
;    0009  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0010  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0011  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0012  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0013  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0014  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0015  0      ! REMAIN IN DEC.
;    0016  0      !
;    0017  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0018  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0019  0      ! CORPORATION.
;    0020  0      !
;    0021  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0022  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0023  0      !
;    0024  0      !++
;    0025  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0026  0      !
;    0027  0      ! ABSTRACT:
;    0028  0      !
;    0029  0      !       This module contains the routines required to SET and SHOW the
;    0030  0      !       various QA flags.
;    0031  0      !
;    0032  0      ! AUTHOR:
;    0033  0      !
;    0034  0      !       Jack Stansbury
;    0035  0      !
;    0036  0      ! CREATION DATE:
;    0037  0      !
;    0038  0      !       25-October-1981
;    0039  0      !
;    0040  0      ! MODIFIED:
;    0041  0      !
;    0042  0      !       01      Jack Stansbury, 21-Nov-1981, Version 6.5
;    0043  0      !               Added INITIAL attributes to the flag word storage.
;    0044  0      !
;    0045  0      !       02      Jack Stansbury, 14-Jan-1982, Version 6.6
;    0046  0      !               Changed the lower limit on the QASUBTESTLOOPS DS flag from
;    0047  0      !               1 to 2. This makes the Loop_On_Subtest routine much easier.
;    0048  0      !               Also changed the Out of Range message to say "greater than or
;    0049  0      !               equal to" rather than just "greater than".
;    0050  0      !
;    0051  0      !       03      Jack Stansbury, 20-Jan-1983, Version 6.11
;    0052  0      !               Took out all references to CkLoopLoops and ErrorPrints.
;    0053  0      !
;    0054  0      !       04      Bob Bergazzi    May 17, 1983    Version 6.11
;    0055  0      !               Changed the order of searching libraries.
;    0056  0      !--
```

```
;    0057  1     BEGIN
;    0058  1     %SBTTL  'Libraries'
;    0059  1     !
;    0060  1     ! LIBRARIES:
;    0061  1     !
;    0062  1
;    0063  1     LIBRARY '$DIAG';                    !                                   [04]
;    0064  1
;    0065  1     LIBRARY '$DS';                      !                                   [04]
;    0066  1
;    0067  1     LIBRARY 'SYS$LIBRARY:LIB';
```

```
;    0068  1     %SBTTL  'Table of Contents'
;    0069  1     !
;    0070  1     ! TABLE OF CONTENTS:
;    0071  1     !
;    0072  1
;    0073  1     FORWARD ROUTINE
;    0074  1             VRSetQA         : JSB_CLI,
;    0075  1             VRShowQA        : JSB_CLI;
```

```
;    0076  1      %SBTTL  'Psect Definitions'
;    0077  1      !
;    0078  1      ! PSECT DEFINITIONS:
;    0079  1      !
;    0080  1
;    0081  1      PSECT
;    0082  1              Plit   = Data (NoExecute, Share, NoWrite,
;    0083  1                       Addressing_Mode (Long_Relative));
;    0084  1
;    0085  1      PSECT
;    0086  1              Code   = Code (Execute, Share, NoWrite);
;    0087  1
;    0088  1      PSECT
;    0089  1              Own    = Work (NoExecute, NoShare, Write,
;    0090  1                       Addressing_Mode (Long_Relative));
;    0091  1
;    0092  1      PSECT
;    0093  1              Global = Work (NoExecute, NoShare, Write,
;    0094  1                       Addressing_Mode (Long_Relative));
```

```
;    0095  1      %SBTTL  'Literals'
;    0096  1
;    0097  1      ! MODULE-GLOBAL LITERALS:
;    0098  1      !
;    0099  1      LITERAL
;    0100  1              QA$K_TestLoops_Def      = 100,          ! Defaults for the 5 QA flags
;    0101  1              QA$K_SubtestLoops_Def   = 100,
;    0102  1              QA$K_MultiplePass_Def   = 10,
;    0103  1
;    0104  1              QA$K_Success            = 1,            ! Routine Success or
;    0105  1              QA$K_Failure            = 0;            !   Failure
;    0106  1
;    0107  1      LITERAL
;    0108  1              !+
;    0109  1              ! Shorthand forms for the QA flags
;    0110  1              !
;    0111  1              !       TL => TestLoops
;    0112  1              !       SL => SubtestLoops
;    0113  1              !       MP => MultiplePass
;    0114  1              !-
;    0115  1
;    0116  1              QATL_Low        = 1,    ! Lower limit of QATESTLOOPS
;    0117  1              QATL_High       = 32767,! Upper limit of QATESTLOOPS
;    0118  1
;    0119  1              QASL_Low        = 2,    ! Lower limit of QASUBTESTLOOPS                 [02]
;    0120  1              QASL_High       = 32767,! Upper limit of QASUBTESTLOOPS
;    0121  1
;    0122  1              QAMP_Low        = 1,    ! Lower limit of QAMULTIPLEPASS
;    0123  1              QAMP_High       = 32767;! Upper limit of QAMULTIPLEPASS
;    0124  1
```

```
;    0125  1      %SBTTL  'Static Storage'
;    0126  1
;    0127  1      ! MODULE-GLOBAL DECLARATIONS:
;    0128  1      !
;    0129  1
;    0130  1      GLOBAL
;    0131  1              !+
;    0132  1              ! For storing the current values of the QA flags.
;    0133  1              !-
;    0134  1
;    0135  1              QA$W_TestLoops          : WORD INITIAL (QA$K_TestLoops_Def),
;    0136  1              QA$W_SubtestLoops       : WORD INITIAL (QA$K_SubtestLoops_Def),
;    0137  1              QA$W_MultiplePass       : WORD INITIAL (QA$K_MultiplePass_Def);
```

```
;    0138  1      %SBTTL  'Global Bindings'
;    0139  1      !
;    0140  1      ! Module-Global Bindings
;    0141  1      !
;    0142  1      BIND
;    0143  1              QAMP_Out = $ASCIC ('QAMULTIPLEPASS'),
;    0144  1              QATL_Out = $ASCIC ('QATESTLOOPS'),
;    0145  1              QASL_Out = $ASCIC ('QASUBTESTLOOPS');
```

```
;    0146  1     %SBTTL 'VRSetQA'
;    0147  1
;    0148  1     GLOBAL ROUTINE VRSetQA (Cli_Base) : JSB_CLI =
;    0149  1
;    0150  1     !++
;    0151  1     ! FUNCTIONAL DESCRIPTION:
;    0152  1     !
;    0153  1     !       This routine is called from the CLI module. It sets the specified
;    0154  1     !       QA flag to the requested value (within limits).
;    0155  1     !
;    0156  1     ! CALLER(S):
;    0157  1     !
;    0158  1     !       CLI
;    0159  1     !
;    0160  1     ! FORMAL PARAMETERS:
;    0161  1     !
;    0162  1     !       R2 - Points to the CLI data table base.
;    0163  1     !
;    0164  1     ! IMPLICIT INPUTS:
;    0165  1     !
;    0166  1     !       Cli_Base [CLI$L_FLAGS] - Indicates which flag to change.
;    0167  1     !       Cli_Base [CLI$L_DATA]  - Indicates the new desired value of the flag.
;    0168  1     !
;    0169  1     ! IMPLICIT OUTPUTS:
;    0170  1     !
;    0171  1     !       New value(s) for the QAx variables declared above.
;    0172  1     !
;    0173  1     ! COMPLETION CODES:
;    0174  1     !
;    0175  1     !       0 - Failure, value out of range.
;    0176  1     !       1 - Success.
;    0177  1     !
;    0178  1     ! SIDE EFFECTS:
;    0179  1     !
;    0180  1     !       Changes the value of the specifed flag, iff the value requested is
;    0181  1     !       within range.
;    0182  1     !--
```

```
;    0183  1
;    0184  2     BEGIN
;    0185  2             MAP
;    0186  2                     Cli_Base : REF BLOCK [, BYTE];
;    0187  2
;    0188  2             LOCAL
;    0189  2                     New_Value;
;    0190  2
;    0191  2             BIND
;    0192  2                     Out_Of_Range_Message =
;    0193  2                             $ASCIC ('!AC must be greater than or equal to !UW and less than or equal to !UW.!/');   !
;    0194  2
;    0195  2             MACRO
;  M 0196  2                     Check_Case (Short_Name, Data_Name, Cli_Name) =
;  M 0197  2                             [.Cli_Base [%NAME ('CLI$V_', Cli_Name)]]:
;  M 0198  2                                     IF (
;  M 0199  2                                        (.New_Value LSS %NAME (Short_Name, '_Low'))
;  M 0200  2                                      OR
;  M 0201  2                                        (.New_Value GTR %NAME (Short_Name, '_High'))
;  M 0202  2                                        )
;  M 0203  2                                     THEN
;  M 0204  2                                             BEGIN
;  M 0205  2                                             $DS_PRINTF (
;  M 0206  2                                                     Out_Of_Range_Message,
;  M 0207  2                                                     %NAME (Short_Name, '_Out'),
;  M 0208  2                                                     %NAME (Short_name, '_Low'),
;  M 0209  2                                                     %NAME (Short_Name, '_High')
;  M 0210  2                                                     );
;  M 0211  2                                             RETURN (QA$K_Failure)
;  M 0212  2                                             END
;  M 0213  2                                     ELSE
;  M 0214  2                                             Data_Name = .New_Value
;    0215  2                             %;
```

```
;    0216  2
;    0217  2              New_Value = .Cli_Base [CLI$L_DATA];
;    0218  2
;    0219  2              SELECTONE 1 OF
;    0220  2                      SET
;    0221  2                      Check_Case (QATL, QA$W_TestLoops,    QATestLoops);
;    0222  2                      Check_Case (QASL, QA$W_SubtestLoops, QASubtestLoops);
;    0223  2                      Check_Case (QAMP, QA$W_MultiplePass, QAMultiplePass);
;    0224  2                      [.Cli_Base [CLI$V_DEFAULT]]:
;    0225  2                              !
;    0226  2                              ! Set all of them to their defaults:
;    0227  2                              !
;    0228  3                              BEGIN
;    0229  3                              QA$W_TestLoops    = QA$K_TestLoops_Def;
;    0230  3                              QA$W_SubtestLoops = QA$K_SubtestLoops_Def;
;    0231  3                              QA$W_MultiplePass = QA$K_MultiplePass_Def
;    0232  3                              END
;    0233  2                      TES;
;    0234  2
;    0235  3              RETURN (QA$K_Success)
;    0236  1      END;


                                                        .TITLE  QAFLAGS *** Setting and Showing QA Flags
                                                        .IDENT  \6.6-04\

                                                        .PSECT  WORK,NOEXE,2

                                              0064  00000 QA$W_TESTLOOPS::
                                                        .WORD   100                                     ;
                                              0064  00002 QA$W_SUBTESTLOOPS::
                                                        .WORD   100                                     ;
                                              000A  00004 QA$W_MULTIPLEPASS::
                                                        .WORD   10                                      ;

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

53 53 41 50 45 4C 50 49 54 4C 55 4D 41 51 0E  00000 P.AAA:   .ASCII  <14>\QAMULTIPLEPASS\                ;
      53 50 4F 4F 4C 54 53 45 54 41 51 0B      0000F P.AAB:   .ASCII  <11>\QATESTLOOPS\                  ;
53 50 4F 4F 4C 54 53 45 54 42 55 53 41 51 0E  0001B P.AAC:   .ASCII  <14>\QASUBTESTLOOPS\               ;
72 67 20 65 62 20 74 73 75 6D 20 43 41 21 49  0002A P.AAD:   .ASCII  \I!AC must be greater than or equal to !U\ ;
65 20 72 6F 20 6E 61 68 74 20 72 65 74 61 65  00039                                                   ;
         55 21 20 6F 74 20 6C 61 75 71         00048                                                   ;
6E 61 68 74 20 73 73 65 6C 20 64 6E 61 20 57  00052           .ASCII  \W and less than or equal to !UW.!/\ ;
55 21 20 6F 74 20 6C 61 75 71 65 20 72 6F 20  00061                                                   ;
                              2F 21 2E 57     00070                                                   ;

                                                        QAMP_OUT=               P.AAA
                                                        QATL_OUT=               P.AAB
                                                        QASL_OUT=               P.AAC
                                                        OUT_OF_RANGE_MESSAGE=
                                                                                P.AAD
                                                        .EXTRN  DS$PRINTF

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                              53  DD 00000 VRSETQA::
```

```
                                                                PUSHL   R3                                              ; 0148
                        53      1C      A2  D0 00002            MOVL    28(CLI_BASE), NEW_VALUE                         ; 0217
                23      62              1D  E1 00006            BBC     #29, (CLI_BASE), 3$                             ; 0221
                                        09  15 0000A            BLEQ    1$                                              ;
                00007FFF 8F             53  D1 0000C            CMPL    NEW_VALUE, #32767                               ;
                                        0F  15 00013            BLEQ    2$                                              ;
                        7E      7FFF    8F  3C 00015 1$:        MOVZWL  #32767, -(SP)                                   ;
                                        01  DD 0001A            PUSHL   #1                                              ;
                            00000000'   EF  9F 0001C            PUSHAB  QATL_OUT                                        ;
                                        51  11 00022            BRB     8$                                              ;
                00000000'   EF          53  B0 00024 2$:        MOVW    NEW_VALUE, QA$W_TESTLOOPS                       ;
                                        76  11 0002B            BRB     11$                                             ;
                26      62              1E  E1 0002D 3$:        BBC     #30, (CLI_BASE), 6$                             ; 0222
                        02              53  D1 00031            CMPL    NEW_VALUE, #2                                   ;
                                        09  19 00034            BLSS    4$                                              ;
                00007FFF 8F             53  D1 00036            CMPL    NEW_VALUE, #32767                               ;
                                        0F  15 0003D            BLEQ    5$                                              ;
                        7E      7FFF    8F  3C 0003F 4$:        MOVZWL  #32767, -(SP)                                   ;
                                        02  DD 00044            PUSHL   #2                                              ;
                            00000000'   EF  9F 00046            PUSHAB  QASL_OUT                                        ;
                                        27  11 0004C            BRB     8$                                              ;
                00000000'   EF          53  B0 0004E 5$:        MOVW    NEW_VALUE, QA$W_SUBTESTLOOPS                    ;
                                        4C  11 00055            BRB     11$                                             ;
                                        62  D5 00057 6$:        TSTL    (CLI_BASE)                                      ; 0223
                                        32  18 00059            BGEQ    10$                                             ;
                                        53  D5 0005B            TSTL    NEW_VALUE                                       ;
                                        09  15 0005D            BLEQ    7$                                              ;
                00007FFF 8F             53  D1 0005F            CMPL    NEW_VALUE, #32767                               ;
                                        1C  15 00066            BLEQ    9$                                              ;
                        7E      7FFF    8F  3C 00068 7$:        MOVZWL  #32767, -(SP)                                   ;
                                        01  DD 0006D            PUSHL   #1                                              ;
                            00000000'   EF  9F 0006F            PUSHAB  QAMP_OUT                                        ;
                            00000000'   EF  9F 00075 8$:        PUSHAB  OUT_OF_RANGE_MESSAGE                            ;
                    00000000G   9F          04  FB 0007B            CALLS   #4, @#OTS$PRINTF                            ;
                                        24  11 00082            BRB     12$                                             ;
                00000000'   EF          53  B0 00084 9$:        MOVW    NEW_VALUE, QA$W_MULTIPLEPASS                    ;
                                        16  11 0008B            BRB     11$                                             ;
                    12      62              0C  E1 0008D 10$:       BBC     #12, (CLI_BASE), 11$                        ; 0224
                00000000'   EF  00640064    8F  D0 00091            MOVL    #6553700, QA$W_TESTLOOPS                    ; 0229
                00000000'   EF              0A  B0 0009C            MOVW    #10, QA$W_MULTIPLEPASS                      ; 0231
                            50              01  D0 000A3 11$:       MOVL    #1, R0                                      ; 0235
                                            02  11 000A6            BRB     13$                                         ;
                                            50  D4 000A8 12$:       CLRL    R0                                          ; 0236
                                            08  BA 000AA 13$:       POPR    #^M<R3>                                     ;
                                                05 000AC            RSB                                                 ;

; Routine Size:  173 bytes,    Routine Base:  CODE + 0000

;   0237  1
```

```
;    0238  1     %SBTTL 'VRShowQA'
;    0239  1
;    0240  1     GLOBAL ROUTINE VRShowQA (Cli_Base) : JSB_CLI =
;    0241  1
;    0242  1     !++
;    0243  1     ! FUNCTIONAL DESCRIPTION:
;    0244  1     !
;    0245  1     !       This global routine is called from the CLI module. It shows the
;    0246  1     !       current value of the specified QA flag.
;    0247  1     !
;    0248  1     ! CALLER(S):
;    0249  1     !
;    0250  1     !       CLI
;    0251  1     !
;    0252  1     ! FORMAL PARAMETERS:
;    0253  1     !
;    0254  1     !       R2 - Points to the CLI data-table base.
;    0255  1     !
;    0256  1     ! IMPLICIT INPUTS:
;    0257  1     !
;    0258  1     !       Cli_Base [CLI$L_FLAGS] - Indicates which flag to ''SHOW''.
;    0259  1     !
;    0260  1     ! IMPLICIT OUTPUTS:
;    0261  1     !
;    0262  1     !       NONE
;    0263  1     !
;    0264  1     ! COMPLETION CODES:
;    0265  1     !
;    0266  1     !       1 - Success.
;    0267  1     !
;    0268  1     ! SIDE EFFECTS:
;    0269  1     !
;    0270  1     !       NONE
;    0271  1     !--
```

```
:    0272  1
:    0273  2     BEGIN
:    0274  2             MAP
:    0275  2                     Cli_Base : REF BLOCK [, BYTE];
:    0276  2
:    0277  2             BIND
:    0278  2                     Output_String   = $ASCIC ('!AC is set to !UW.!/'),
:    0279  2                     Default_String  = $ASCIC ('The default for !AC is !UW.!/');
:    0280  2
:    0281  2
:    0282  2             MACRO
: M  0283  2                     Check_Case (Short_Name, Data_Name, Cli_Name) =
: M  0284  2                             [.Cli_Base [%NAME ('CLI$V_', Cli_Name)]]:
: M  0285  2                                     $DS_PRINTF (Output_String,
: M  0286  2                                                 %NAME (Short_Name, '_Out'),
: M  0287  2                                                 .Data_Name)
:    0288  2                     %;
```

```
;    0289  2
;    0290  2              SELECTONE 1 OF
;    0291  2                      SET
;    0292  2                      Check_Case (QATL, QA$W_TestLoops,    QATestLoops);
;    0293  2                      Check_Case (QASL, QA$W_SubtestLoops, QASubtestLoops);
;    0294  2                      Check_Case (QAMP, QA$W_MultiplePass, QAMultiplePass);
;    0295  2                      [.Cli_Base [CLI$V_DEFAULT]]:
;    0296  3                              BEGIN
;  P 0297  3                              $DS_PRINTF (Default_String, QAMP_Out,
;    0298  3                                          QA$K_MultiplePass_Def);
;    0299  3
;  P 0300  3                              $DS_PRINTF (Default_String, QATL_Out,
;    0301  3                                          QA$K_TestLoops_Def);
;    0302  3
;  P 0303  3                              $DS_PRINTF (Default_String, QASL_Out,
;    0304  4                                          QA$K_SubtestLoops_Def)
;    0305  2                              END;
;    0306  2                      TES;
;    0307  2
;    0308  3              RETURN (QA$K_Success)
;    0309  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20 6F 74 20 74 65 73 20 73 69 20 43 41 21 14  00074 P.AAE:  .ASCII  <20>\!AC is set to !UW.!/\                    ;
                                              2F 21 2E 57 55 21 00083                                             ;
6F 66 20 74 6C 75 61 66 65 64 20 65 68 54 1D  00089 P.AAF:  .ASCII  <29>\The default for !AC is !UW.!/\           ;
2F 21 2E 57 55 21 20 73 69 20 43 41 21 20 72  00098                                                             ;

                                                                OUTPUT_STRING=      P.AAE
                                                                DEFAULT_STRING=     P.AAF


                                                                .PSECT  CODE,NOWRT,  SHR,2

                    0F              62              1D  E1 00000 VRSHOWQA::
                                                                  BBC     #29, (CLI_BASE), 1$                   ; 0292
                                    7E 00000000'    EF  3C 00004        MOVZWL  QA$W_TESTLOOPS, -(SP)             ;
                                       00000000'    EF  9F 0000B        PUSHAB  QATL_OUT                          ;
                                                    24  11 00011        BRB     3$                                ;
                    0F              62              1E  E1 00013 1$:    BBC     #30, (CLI_BASE), 2$               ; 0293
                                    7E 00000000'    EF  3C 00017        MOVZWL  QA$W_SUBTESTLOOPS, -(SP)          ;
                                       00000000'    EF  9F 0001E        PUSHAB  QASL_OUT                          ;
                                                    11  11 00024        BRB     3$                                ;
                                                    62  D5 00026 2$:    TSTL    (CLI_BASE)                        ; 0294
                                                    15  18 00028        BGEQ    4$                                ;
                                    7E 00000000'    EF  3C 0002A        MOVZWL  QA$W_MULTIPLEPASS, -(SP)          ;
                                       00000000'    EF  9F 00031        PUSHAB  QAMP_OUT                          ;
                                       00000000'    EF  9F 00037 3$:    PUSHAB  OUTPUT_STRING                     ;
                                                    40  11 0003D        BRB     5$                                ;
                    43              62              0C  E1 0003F 4$:    BBC     #12, (CLI_BASE), 6$               ; 0295
                                                    0A  DD 00043        PUSHL   #10                               ; 0298
                                       00000000'    EF  9F 00045        PUSHAB  QAMP_OUT                          ;
                                       00000000'    EF  9F 0004B        PUSHAB  DEFAULT_STRING                    ;
                        00000000G   9F              03  FB 00051        CALLS   #3, @#DS$PRINTF                   ;
```

```
                                 7E       64   8F  9A 00058          MOVZBL  #100, -(SP)                                 ; 0301
                              00000000'       EF  9F 0005C          PUSHAB  QATL_OUT                                    ;
                              00000000'       EF  9F 00062          PUSHAB  DEFAULT_STRING                              ;
                    00000000G 9F              03  FB 00068          CALLS   #3, a#DS$PRINTF                             ;
                                 7E       64   8F  9A 0006F          MOVZBL  #100, -(SP)                                 ; 0304
                              00000000'       EF  9F 00073          PUSHAB  QASL_OUT                                    ;
                              00000000'       EF  9F 00079          PUSHAB  DEFAULT_STRING                              ;
                    00000000G 9F              03  FB 0007F 5$:       CALLS   #3, a#DS$PRINTF                             ;
                                 50           01  D0 00086 6$:       MOVL    #1, R0                                      ; 0308
                                              05  00089              RSB                                                 ; 0309

; Routine Size:  138 bytes,    Routine Base:  CODE + 00AD
```

```
;   0310  1     %SBTTL 'End of Module QAFLAGS'
;   0311  1
;   0312  0     END ELUDOM
```

```
;                                PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  WORK                             6  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  DATA                           167  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                           311  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                              Library Statistics

;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time

;  DRB1:[DS.WORK]DIAG.L32;265          784         2         0          85        00:00.2
;  DRB1:[DS.WORK]DS.L32;159            653         6         0          42        00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7     18017         0         0         975        00:04.7
```

```
;                                  COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE QAFLAGS

; Size:          311 code + 173 data bytes
; Run Time:        00:11.4
; Elapsed Time:    00:33.5
; Lines/CPU Min:    1649
; Lexemes/CPU-Min: 10007
; Memory Used:  84 pages
; Compilation Complete
```

```
;    0001  0     %TITLE '*** QA Routines'
;    0002  0     MODULE QAMAIN (
;    0003  0               ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE),
;    0004  0               IDENT = '1-08'
;    0005  0               ) =
;    0006  0     !
;    0007  0     ! COPYRIGHT (c) 1979, 1981, 1983
;    0008  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0009  0     !
;    0010  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0011  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0012  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0013  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0014  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0015  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0016  0     ! REMAIN IN DEC.
;    0017  0     !
;    0018  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0019  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0020  0     ! CORPORATION.
;    0021  0     !
;    0022  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0023  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0024  0     !
;    0025  0     !++
;    0026  0     ! FACILITY:
;    0027  0     !
;    0028  0     !       DIAGNOSTIC SUPERVISOR
;    0029  0     !
;    0030  0     ! ABSTRACT:
;    0031  0     !
;    0032  0     !       This module contains the support routines required to perform QA on diagnostic programs.
;    0033  0     !
;    0034  0     ! AUTHOR:
;    0035  0     !
;    0036  0     !       Jack Stansbury
;    0037  0     !
;    0038  0     ! CREATION DATE:
;    0039  0     !
;    0040  0     !               15-October-1981, Version 6.5-00
;    0041  0     !       00      Put in code for all the support routines plus the following check routines: QA$Normal_Start,
;    0042  0     !               QA$Multiple_Pass, QA$Loop_On_Test, QA$Run_Backwards.
```

```
;    0043  0      ! MODIFIED:
;    0044  0      !
;    0045  0      !       01      Jack Stansbury, 14-December-1981, Version 6.6
;    0046  0      !               Added remaining QA check routines. Split QA.B32 into two different modules: QAMAIN.B32
;    0047  0      !               (this one) and QACHECKS.B32. While this one contains the support routines, the latter contains
;    0048  0      !               the actual check routines.
;    0049  0      !
;    0050  0      !       02      Jack Stansbury, 5-Jan-1982, Version 6.6
;    0051  0      !               Changed the code in QA$Dump to not print R0 and R1 in the register dump. There is no need to
;    0052  0      !               print these since they are not saved by the CALLx instructions.
;    0053  0      !
;    0054  0      !       03      Jack Stansbury, 1-Feb-1982, Version 6.6
;    0055  0      !               Released the three Supervisors to BSDE to find bugs. This includes all the check routine up to
;    0056  0      !               and including the Error Phase One check.
;    0057  0      !
;    0058  0      !       04      Jack Stansbury, 1-Feb-1982, Version 6.6
;    0059  0      !               Changed the headings on the two table printouts to make the table name more distinctive.
;    0060  0      !
;    0061  0      !       05      Jack Stansbury, 11-Feb-1982, Version 6.6X
;    0062  0      !               Changed the DSX$Branch routine so as to make it compatible with Sam Duncan! He wanted R0 saved
;    0063  0      !               across the branch macro code expansion, so I made significant changes to both the branch macros
;    0064  0      !               (in DIAG.*) and to this branch routine. All of that just for Sam!
;    0065  0      !
;    0066  0      !       06      Jack Stansbury, 27-July-1982, Version 6.9
;    0067  0      !               Changed the spelling of the VDS ^C handler variable.
;    0068  0      !
;    0069  0      !       07      Jack Stansbury, 19-Jan-1983, Version 6.10 (?)
;    0070  0      !               Took out all the references to any of the checks above the Run Backwards check. This
;    0071  0      !               includes taking out the previously entered Error Phase One check. It doesn't work
;    0072  0      !               properly with subroutines in diagnostic programs.
;    0073  0      !
;    0074  0      !       08      Bob Bergazzi    May 17, 1983    Version 6.11
;    0075  0      !               Changed the order of searching libraries.
;    0076  0      !--
```

```
;    0077  0     %SBTTL  'Linkage Definitions'
;    0078  1     BEGIN
;    0079  1     !
;    0080  1     ! LINKAGE DEFINITIONS
;    0081  1     !
;    0082  1
;    0083  1     LINKAGE
;    0084  1             !+
;    0085  1             ! For the DSX$Branch routine, the last three parameters are passed in the STANDARD manner. That is      [05]
;    0086  1             ! by an offset to the AP on the stack. The first parameter passed is general register R0. Also,        [05]
;    0087  1             ! register R1 should be preserved in the routine. DSX$Branch is defined to be a VALUE routine also, so [05]
;    0088  1             ! that values can be returned in R0. This is not compatible with the VAX calling standard however.    [05]
;    0089  1             !-
;    0090  1             Call_DSX$Branch = CALL (REGISTER=0, STANDARD, STANDARD, STANDARD) : PRESERVE (1);                    !      [05]
```

```
;    0091  1     %SBTTL  'Table of Contents'
;    0092  1     !
;    0093  1     ! TABLE OF CONTENTS:
;    0094  1     !
;    0095  1
;    0096  1     FORWARD ROUTINE
;    0097  1             DSX$Branch                  :                   ADDRESSING_MODE (LONG_RELATIVE) Call_DSX$Branch.        !       [05]
;    0098  1             QA$Abort_QA                 :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),        !               [02]
;    0099  1             QA$Add_Forced_Error         :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),        !               [02]
;    0100  1             QA$Add_QA_Error             :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),
;    0101  1             QA$Control_C_Handler        :                   ADDRESSING_MODE (LONG_RELATIVE),        !               [06]
;    0102  1             QA$Dump                     :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),
;    0103  1             QA$Find_User_Call_Frame     :                   ADDRESSING_MODE (LONG_RELATIVE),
;    0104  1             QA$Init                     :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),
;    0105  1             QA$Main                     :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),
;    0106  1             QA$Print_Forced_Errors      :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE),        !               [02]
;    0107  1             QA$Print_Overall_Errors     :       NOVALUE     ADDRESSING_MODE (LONG_RELATIVE);        !               [02]
```

```
:    0108  1     %SBTTL  'Libraries'
:    0109  1     !
:    0110  1     ! LIBRARIES:
:    0111  1     !
:    0112  1
:    0113  1     LIBRARY
:    0114  1             '$DIAG';                !                                   [08]
:    0115  1
:    0116  1     LIBRARY
:    0117  1             '$DS';                  !                                   [08]
:    0118  1
:    0119  1     LIBRARY
:    0120  1             'SYS$LIBRARY:LIB';
```

```
:    0121  1     %SBTTL  'Included Macros and Literals'
:    0122  1     !
:    0123  1     ! INCLUDED MACROS AND LITERALS
:    0124  1     !
:    0125  1
:    0126  1             $DS_QADEF;                      ! Define the branch-codes.
:    0127  1             $DS_DSADEF;                     ! Define DSA flags
:    0128  1             DSQA;                           ! The DSQA$K_ constants for the routine/label names.
:    0129  1             DS_QADEFS;                      ! Common QA definitions like QA$K_.
```

```
;    0130  1     %SBTTL  'External Routines'
;    0131  1     !
;    0132  1     ! EXTERNAL ROUTINE REFERENCES:
;    0133  1     !
;    0134  1
;    0135  1     EXTERNAL ROUTINE
;    0136  1
;    0137  1             QA$Normal_Start         :               ADDRESSING_MODE (LONG_RELATIVE),
;    0138  1             QA$Multiple_Pass        :               ADDRESSING_MODE (LONG_RELATIVE),
;    0139  1             QA$Loop_On_Test         :               ADDRESSING_MODE (LONG_RELATIVE),
;    0140  1             QA$Loop_On_Subtest      :               ADDRESSING_MODE (LONG_RELATIVE),        !                       [01]
;    0141  1             QA$Run_Backwards        :               ADDRESSING_MODE (LONG_RELATIVE),
;    0142  1
;    0143  1             EXE$AloNonPaged         : JSB_ALLOC     ADDRESSING_MODE (LONG_RELATIVE),        !                       [01]
;    0144  1             EXE$DeaNonPaged         : JSB_DEALL     ADDRESSING_MODE (LONG_RELATIVE),        !                       [01]
;    0145  1
;    0146  1             DSV$ShowDevice          : JSB_CLI,              ! Pass parameter in R2
;    0147  1             DSV$ShowSelect          : JSB_CLI,              ! Pass parameter in R2
;    0148  1             DSX$Abort,                                      ! Pass no parameters,
;    0149  1                                                             ! (take no prisoners).
;    0150  1             VRShowFlg               : JSB_NONE,             ! No parameters required
;    0151  1             VRShowEf                : JSB_NONE;             ! No parameters required
```

```
;    0152  1      %SBTTL  'External Own Storage'
;    0153  1      !
;    0154  1      ! EXTERNAL OWN STORAGE
;    0155  1      !
;    0156  1
;    0157  1      EXTERNAL
;    0158  1              DS$GL_FSTTEST,                          ! First test to execute
;    0159  1              DS$GL_LSTTEST,                          ! Last test to execute
;    0160  1
;    0161  1              DS$GA_DS_Ctrl_C_First,                  ! DS address of 1st ^C routine handler                  [06]
;    0162  1
;    0163  1              DS$GL_CLIBASE,                          ! Base of CLI data table
;    0164  1
;    0165  1              DS$GB_WIDTH,                            ! Width of the terminal                                 [02]
;    0166  1
;    0167  1              !+
;    0168  1              ! These are defined in the QAFLAGS module. They contain
;    0169  1              ! the current settings of the QA flags.
;    0170  1              !-
;    0171  1
;    0172  1              QA$W_TestLoops    : WORD,
;    0173  1              QA$W_SubtestLoops : WORD,
;    0174  1              QA$W_MultiplePass : WORD,
;    0175  1
;    0176  1              QA$T_Header_1,                          ! The three lines printed at the top                    [01]
;    0177  1              QA$T_Header_2,                          ! of a page.                                            [01]
;    0178  1              QA$T_Header_3,                          !                                                       [01]
;    0179  1
;    0180  1              QA$T_Operator_Request_Message;          ! The QA error message for requesting operator input    [02]
;    0181  1                                                     ! with no default value given. It is declared in the    [02]
;    0182  1                                                     ! QACHECKS module because all the error messages are    [02]
;    0183  1                                                     ! declared there!                                       [02]
```

```
;    0184  1     %SBTTL  'Psect Definitions'
;    0185  1     !
;    0186  1     ! PSECT DEFINITIONS:
;    0187  1     !
;    0188  1
;    0189  1     PSECT
;    0190  1             Plit   = Data (NoExecute, Share, NoWrite,
;    0191  1                            Addressing_Mode (Long_Relative));
;    0192  1
;    0193  1     PSECT
;    0194  1             Code   = Code (Execute, Share, NoWrite,
;    0195  1                            Addressing_Mode (Long_Relative));
;    0196  1
;    0197  1     PSECT
;    0198  1             Own    = Work (NoExecute, NoShare, Write,
;    0199  1                            Addressing_Mode (Long_Relative));
;    0200  1
;    0201  1     PSECT
;    0202  1             Global = Work (NoExecute, NoShare, Write,
;    0203  1                            Addressing_Mode (Long_Relative));
```

```
;    0204  1      %SBTTL  'Local Macro Definitions'
;    0205  1      !
;    0206  1      ! LOCAL MACRO DEFINITIONS:
;    0207  1      !
;    0208  1
;    0209  1      MACRO
;    0210  1              !+
;    0211  1              ! Usage:
;    0212  1              !       ASSERT ((.A GTR .B), 'A <= B');
;    0213  1              !
;    0214  1              ! If the specified condition is not true, a call to the Logic_Error macro is made with the specified
;    0215  1              ! message string. ASSERT is turned on whenever the compilation is made with the /VARIANT:n qualifier,
;    0216  1              ! where n <> 0.
;    0217  1              !-
;    0218  1
;  M 0219  1              ASSERT (Condition, Assert_String) =
;  M 0220  1                      %IF %VARIANT
;  M 0221  1                      %THEN
;  M 0222  1                              (
;  M 0223  1                              IF (NOT (Condition)) THEN
;  M 0224  1                                      Logic_Error (Assert_String)
;  M 0225  1                              )
;  M 0226  1                      %ELSE
;  M 0227  1                              ! No code put in.
;  M 0228  1                      %FI
;    0229  1              %;
;    0230  1
;    0231  1      MACRO
;    0232  1              !+
;    0233  1              ! Usage:
;    0234  1              !       Logic_Error ('String to be printed out');
;    0235  1              !
;    0236  1              ! This macro is used whenever a logic error is detected. It is called from ASSERT whenever the
;    0237  1              ! assertion is false. It can also be called whenever an expression need not be evaluated before
;    0238  1              ! determining that something is wrong.
;    0239  1              !-
;    0240  1
;  M 0241  1              Logic_Error (Error_String) =
;  M 0242  1                      (
;  M 0243  1                      BIND
;  M 0244  1                              ES = $ASCIC ('!/**QA Internal Error: ', Error_String, '!/');    !                       [01]
;  M 0245  1
;  M 0246  1                      $DS_PRINTF (ES);
;  M 0247  1                      )
;    0248  1              %;
;    0249  1
;    0250  1      MACRO                                                   !                                                       [02]
;    0251  1              !+                                                                                                      [02]
;    0252  1              ! Define a macro that will expand into the declaration for the Forced-Error records.                    [02]
;    0253  1              !-                                                                                                      [02]
;    0254  1
;  M 0255  1              Error_Record =                                  !                                                       [02]
;  M 0256  1                      REF BLOCK [, LONG] FIELD (Forced_Error_Fields)  !                                               [02]
;    0257  1              %;                                              !                                                       [02]
```

```
;    0258  1     MACRO
;    0259  1             !+
;    0260  1             ! Usage:
;    0261  1             !
;    0262  1             ! This macro is used only in the debugging stages. It's only parameter is a string giving, e.g.,
;    0263  1             ! the name of a routine. DEBUG is turned on whenever the compilation is made with the
;    0264  1             ! /VARIANT:n qualifier, where n <> 0.
;    0265  1             !-
;    0266  1
; M  0267  1             DEBUG (Debug_String) =
; M  0268  1                     %IF %VARIANT
; M  0269  1                     %THEN
; M  0270  1                             (
; M  0271  1                             IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN
; M  0272  1                                     $DS_PRINTF ($ASCIC ('!/', Debug_String, '!/'))
; M  0273  1                             )
; M  0274  1                     %ELSE
; M  0275  1                             ! No code put in.
; M  0276  1                     %FI
;    0277  1             %;
;    0278  1
;    0279  1     MACRO                                                   !                                       [02]
;    0280  1             !+                                                                                      [02]
;    0281  1             ! Define a macro that takes an expression as an argument. The value of this macro is either one or      [02]
;    0282  1             ! zero. All bits but the least significant one are cleared.                             [02]
;    0283  1             !-                                                                                      [02]
;    0284  1
; M  0285  1             Boolean (Expression) =                          !                                       [02]
; M  0286  1                     ((Expression) AND 1)                    !                                       [02]
;    0287  1             %;                                              !                                       [02]
```

```
;    0288  1     %SBTTL  'Module-Global Literals'
;    0289  1
;    0290  1     !+                                                                                          [02]
;    0291  1     ! MODULE-GLOBAL LITERALS                                                                    [02]
;    0292  1     !-                                                                                          [02]
;    0293  1
;    0294  1     LITERAL                                                 !                                   [02]
;    0295  1             QA$K_Nil = 0,                                   ! Pascal NIL. Use address zero, since it is inacessible.[02]
;    0296  1
;    0297  1             QA$K_Forced_Error_Record_Size = 16,             ! The size of the record for the forced-error records.  [02]
;    0298  1
;    0299  1             QA$K_Last_Check = QA$K_Run_Backwards;           ! The last check currently implemented. This must       [07]
;    0300  1                                                             ! change as more check routines are added.              [07]
```

```
;    0301  1     %SBTTL  'Field Definitions'
;    0302  1
;    0303  1     !                                                                                    [02]
;    0304  1     ! MODULE-GLOBAL FIELD DEFINITIONS                                                    [02]
;    0305  1     !                                                                                    [02]
;    0306  1
;    0307  1     FIELD                                                                                [02]
;    0308  1             Forced_Error_Fields =                                                        [02]
;    0309  1                     SET                                                                  [02]
;    0310  1                     QA$VW_Error_Number    = [0, 0,  16, 0],                              [02]
;    0311  1                     QA$VW_Test_Number     = [0, 16, 16, 0],                              [02]
;    0312  1                     QA$VL_Error_Count     = [1, 0,  32, 0],                              [02]
;    0313  1                     QA$VL_Subtest_Number  = [2, 0,  32, 0],                              [02]
;    0314  1                     Qa$VA_Next_Error      = [3, 0,  32, 0]                               [02]
;    0315  1                     TES;                                                                 [02]
```

```
;    0316  1      %SBTTL  'Module-Global Static Storage'
;    0317  1
;    0318  1      !
;    0319  1      ! MODULE-GLOBAL DECLARATIONS:
;    0320  1      !
;    0321  1
;    0322  1              MODNAM ('QAMAIN');                              ! Define module name.
;    0323  1      OWN
;    0324  1              QA$AW_Overall_Error_Table : VECTOR [QA$K_Number_Of_Checks, WORD];
;    0325  1                                                              ! A vector containing the number of
;    0326  1                                                              ! errors encountered in each of
;    0327  1                                                              ! the checks.
;    0328  1
;    0329  1      !       QA$A_Error_List_Head,                           ! Pointer to head of a linked list of Forced-Error      [07]
;    0330  1      !                                                       ! records.                                              [07]
;    0331  1      !
;    0332  1      !       QA$A_Prev_Test_Tail;                            ! Pointer to the previous test's last record in the     [07]
;    0333  1      !                                                       ! Forced-Error linked list.                             [07]
```

```
;    0334  1     %SBTTL  'Supervisor-Global Static Storage'
;    0335  1
;    0336  1     !+
;    0337  1     ! SUPERVISOR-GLOBAL DECLARATIONS:
;    0338  1     !-
;    0339  1
;    0340  1     GLOBAL
;    0341  1             QA$W_Branch       : WORD,                    ! Storage for passing the branch-code                [01]
;    0342  1                                                          ! from DSX$BRANCH to QA$MAIN.                        [01]
;    0343  1
;    0344  1             QA$W_Error_Number : WORD,                    ! Storage for passing the error number               [01]
;    0345  1                                                          ! from DSX$BRANCH to QA$MAIN.                        [01]
;    0346  1
;    0347  1             QA$W_Save_Test    : SIGNED WORD,             ! Save the x in /TEST:x:y from the command line.     [02]
;    0348  1
;    0349  1             QA$W_Save_Last    : SIGNED WORD,             ! Save the y in /TEST:x:y from the command line.     [02]
;    0350  1
;    0351  1             QA$L_Saved_DSA_Flags,                        ! Save the DSA flags before starting QA              [01]
;    0352  1                                                          ! and restore and end of QA execution.               [01]
;    0353  1
;    0354  1             QA$AOB_Routine_State : BITVECTOR [QA$K_Numb_Routine_States],    !                                [02]
;    0355  1                                                          ! The Routine State vector. This vector keeps track  [02]
;    0356  1                                                          ! of the current state that a check routine is in.   [02]
;    0357  1
;    0358  1             QA$AOB_Check_State : BITVECTOR [QA$K_Number_Of_Checks],
;    0359  1                                                          ! The check state bitvector. It contains one and only
;    0360  1                                                          ! one bit = 1 implying that the check bound to that bit
;    0361  1                                                          ! position is currently being performed.
;    0362  1
;    0363  1             QA$AOB_Flags      : BITVECTOR [QA$K_Number_QA_Flags];
;    0364  1                                                          ! QA$AOB_Flags [QA$K_NoDef] =
;    0365  1                                                          !   1 => No default was given for ASK...
;    0366  1                                                          !   0 => A default was given for ASK...
;    0367  1                                                          !
;    0368  1                                                          ! QA$AOB_Flags [QA$K_QA_Debug] =
;    0369  1                                                          !   1 => Debugging is turned on
;    0370  1                                                          !   0 => No debugging messages
;    0371  1                                                          !
;    0372  1                                                          ! QA$AOB_Flags [QA$K_QA_Done] =
;    0373  1                                                          !   1 => QA is finished
;    0374  1                                                          !   0 => QA is not finished
;    0375  1                                                          !
;    0376  1                                                          ! QA$AOB_Flags [QA$K_Init_Context_Done] =
;    0377  1                                                          !   1 => Init_Context was done
;    0378  1                                                          !   0 => Init_Context has not been done
;    0379  1                                                          !
;    0380  1                                                          ! QA$AOB_Flags [QA$K_Loop_Test_Done] =
;    0381  1                                                          !   1 => Finished looping on the current test
;    0382  1                                                          !   0 => Loop at least once more on the current test
```

```
:    0383  1     %SBTTL 'DSX$BRANCH'
:    0384  1
:    0385  1     GLOBAL ROUTINE DSX$Branch (Register_0, Branch_Code, Error_Number, Branch_Value) : Call_DSX$Branch =
:    0386  1
:    0387  1     !++
:    0388  1     ! FUNCTIONAL DESCRIPTION:
:    0389  1     !
:    0390  1     !       This global routine is called by the $DS_Branch macros. It decides if the branch should be taken.
:    0391  1     !       If the QA flag is set, it will call QA$Main, which calls the appropriate check routine. NOTE:
:    0392  1     !       This routine preserves R1 and attempts to preserve R0 when not forcing an error. At most, the low
:    0393  1     !       bit of R0 will be changed.
:    0394  1     !
:    0395  1     ! FORMAL PARAMETERS:
:    0396  1     !
:    0397  1     !       Register_0    : The general register R0.
:    0398  1     !
:    0399  1     !       Branch_Code   : A literal defining what type of branch is being executed by the diagnostic.
:    0400  1     !
:    0401  1     !       Error_Number  : A diagnostic error number.
:    0402  1     !
:    0403  1     !       Branch_Value  : The value that will determine whether or not the branch should actually be taken.
:    0404  1     !
:    0405  1     ! CALLER(S):
:    0406  1     !
:    0407  1     !       The diagnostic program.
:    0408  1     !
:    0409  1     ! IMPLICIT INPUTS:
:    0410  1     !
:    0411  1     !       NONE
:    0412  1     !
:    0413  1     ! IMPLICIT OUTPUTS:
:    0414  1     !
:    0415  1     !       NONE
:    0416  1     !
:    0417  1     ! COMPLETION CODES:
:    0418  1     !
:    0419  1     !       See below.                                                                              [05]
:    0420  1     !
:    0421  1     ! SIDE EFFECTS:
:    0422  1     !
:    0423  1     !       If running QA and this branch should be reversed, the low bit of R0 will be cleared. If not running QA, [05]
:    0424  1     !       the low bit of R0 will be set or cleared as mandated by the type of branch and by the Branch_Value.     [05]
:    0425  1     !       Under all other conditions, R0 will be left as is. Note: Register_0, the first parameter, is actually   [05]
:    0426  1     !       general register R0.
:    0427  1     !
:    0428  1     !--
```

```
;    0429  2      BEGIN
;    0430  2              LOCAL
;    0431  2                      Save_Register_0,                                        !                    [05]
;    0432  2                      Branch : BYTE;                                          !                    [05]
;    0433  2
;    0434  2              Save_Register_0 = .Register_0;                                  !                    [05]
;    0435  2
;    0436  2              CASE .Branch_Code
;    0437  2                  FROM QA$K_First_Branch_Code TO QA$K_Last_Branch_Code OF
;    0438  2                      SET
;    0439  2                      !+
;    0440  2                      ! These are the BLISS specific branch codes.
;    0441  2                      !-
;    0442  2                      [DS$K_BLSS_B] :
;    0443  2                              Branch = (.Branch_Value LSS 0);
;    0444  2
;    0445  2                      [DS$K_BLSSU_B] :
;    0446  2                              Branch = (.Branch_Value LSSU 0);
;    0447  2
;    0448  2                      [DS$K_BLEQ_B] :
;    0449  2                              Branch = (.Branch_Value LEQ 0);
;    0450  2
;    0451  2                      [DS$K_BLEQU_B] :
;    0452  2                              Branch = (.Branch_Value LEQU 0);
;    0453  2
;    0454  2                      [DS$K_BEQL_B] :
;    0455  2                              Branch = (.Branch_Value EQL 0);
;    0456  2
;    0457  2                      [DS$K_BEQLU_B] :
;    0458  2                              Branch = (.Branch_Value EQLU 0);
;    0459  2
;    0460  2                      [DS$K_BGEQ_B] :
;    0461  2                              Branch = (.Branch_Value GEQ 0);
;    0462  2
;    0463  2                      [DS$K_BGEQU_B] :
;    0464  2                              Branch = (.Branch_Value GEQU 0);
;    0465  2
;    0466  2                      [DS$K_BGTR_B] :
;    0467  2                              Branch = (.Branch_Value GTR 0);
;    0468  2
;    0469  2                      [DS$K_BGTRU_B] :
;    0470  2                              Branch = (.Branch_Value GTRU 0);
;    0471  2
;    0472  2                      [DS$K_BNEQ_B] :
;    0473  2                              Branch = (.Branch_Value NEQ 0);
;    0474  2
;    0475  2                      [DS$K_BNEQU_B] :
;    0476  2                              Branch = (.Branch_Value NEQU 0);
```

```
;    0477  2                    !+
;    0478  2                    ! For these MACRO specific branch codes, the Branch_Value is the PSL at the time
;    0479  2                    ! of the call. "To branch or not to branch" is based on this PSL.
;    0480  2                    !-
;    0481  2
;    0482  2                    [DS$K_BLSS_M] :
;    0483  3                            BEGIN
;    0484  3                            MAP
;    0485  3                                    Branch_Value : BLOCK [, BYTE];
;    0486  3
;    0487  3                            Branch = .Branch_Value [PSL$V_N]
;    0488  2                            END;
;    0489  2
;    0490  2                    [DS$K_BLSSU_M] :
;    0491  3                            BEGIN
;    0492  3                            MAP
;    0493  3                                    Branch_Value : BLOCK [, BYTE];
;    0494  3
;    0495  3                            Branch = .Branch_Value [PSL$V_C]
;    0496  2                            END;
;    0497  2
;    0498  2                    [DS$K_BLEQ_M] :
;    0499  3                            BEGIN
;    0500  3                            MAP
;    0501  3                                    Branch_Value : BLOCK [, BYTE];
;    0502  3
;    0503  3                            Branch = .Branch_Value [PSL$V_N] OR
;    0504  3                                     .Branch_Value [PSL$V_Z]
;    0505  2                            END;
;    0506  2
;    0507  2                    [DS$K_BLEQU_M] :
;    0508  3                            BEGIN
;    0509  3                            MAP
;    0510  3                                    Branch_Value : BLOCK [, BYTE];
;    0511  3
;    0512  3                            Branch = .Branch_Value [PSL$V_C] OR
;    0513  3                                     .Branch_Value [PSL$V_Z]
;    0514  2                            END;
;    0515  2
;    0516  2                    [DS$K_BEQL_M] :
;    0517  3                            BEGIN
;    0518  3                            MAP
;    0519  3                                    Branch_Value : BLOCK [, BYTE];
;    0520  3
;    0521  3                            Branch = .Branch_Value [PSL$V_Z]
;    0522  2                            END;
```

```
;    0523  2                    [DS$K_BEQLU_M] :
;    0524  3                            BEGIN
;    0525  3                            MAP
;    0526  3                                    Branch_Value : BLOCK [, BYTE];
;    0527  3
;    0528  3                            Branch = .Branch_Value [PSL$V_Z]
;    0529  2                            END;
;    0530  2
;    0531  2                    [DS$K_BGEQ_M] :
;    0532  3                            BEGIN
;    0533  3                            MAP
;    0534  3                                    Branch_Value : BLOCK [, BYTE];
;    0535  3
;    0536  3                            Branch = NOT .Branch_Value [PSL$V_N]
;    0537  2                            END;
;    0538  2
;    0539  2                    [DS$K_BGEQU_M] :
;    0540  3                            BEGIN
;    0541  3                            MAP
;    0542  3                                    Branch_Value : BLOCK [, BYTE];
;    0543  3
;    0544  3                            Branch = NOT .Branch_Value [PSL$V_C]
;    0545  2                            END;
;    0546  2
;    0547  2                    [DS$K_BGTR_M] :
;    0548  3                            BEGIN
;    0549  3                            MAP
;    0550  3                                    Branch_Value : BLOCK [, BYTE];
;    0551  3
;    0552  3                            Branch = (NOT .Branch_Value [PSL$V_N]) OR
;    0553  4                                     (NOT .Branch_Value [PSL$V_Z])
;    0554  2                            END;
;    0555  2
;    0556  2                    [DS$K_BGTRU_M] :
;    0557  3                            BEGIN
;    0558  3                            MAP
;    0559  3                                    Branch_Value : BLOCK [, BYTE];
;    0560  3
;    0561  3                            Branch = (NOT .Branch_Value [PSL$V_C]) OR
;    0562  4                                     (NOT .Branch_Value [PSL$V_Z])
;    0563  2                            END;
;    0564  2
;    0565  2                    [DS$K_BNEQ_M] :
;    0566  3                            BEGIN
;    0567  3                            MAP
;    0568  3                                    Branch_Value : BLOCK [, BYTE];
;    0569  3
;    0570  3                            Branch = NOT .Branch_Value [PSL$V_Z]
;    0571  2                            END;
```

```
;    0572  2                    [DS$K_BNEQU_M] :
;    0573  3                            BEGIN
;    0574  3                            MAP
;    0575  3                                    Branch_Value : BLOCK [, BYTE];
;    0576  3
;    0577  3                            Branch = NOT .Branch_Value [PSL$V_Z]
;    0578  2                            END;
;    0579  2
;    0580  2                    [DS$K_BVS_M] :
;    0581  3                            BEGIN
;    0582  3                            MAP
;    0583  3                                    Branch_Value : BLOCK [, BYTE];
;    0584  3
;    0585  3                            Branch = .Branch_Value [PSL$V_V]
;    0586  2                            END;
;    0587  2
;    0588  2                    [DS$K_BVC_M] :
;    0589  3                            BEGIN
;    0590  3                            MAP
;    0591  3                                    Branch_Value : BLOCK [, BYTE];
;    0592  3
;    0593  3                            Branch = NOT .Branch_Value [PSL$V_V]
;    0594  2                            END;
;    0595  2
;    0596  2                    [DS$K_BCS_M] :
;    0597  3                            BEGIN
;    0598  3                            MAP
;    0599  3                                    Branch_Value : BLOCK [, BYTE];
;    0600  3
;    0601  3                            Branch = .Branch_Value [PSL$V_C]
;    0602  2                            END;
;    0603  2
;    0604  2                    [DS$K_BCC_M] :
;    0605  3                            BEGIN
;    0606  3                            MAP
;    0607  3                                    Branch_Value : BLOCK [, BYTE];
;    0608  3
;    0609  3                            Branch = NOT .Branch_Value [PSL$V_C]
;    0610  2                            END;
;    0611  2
;    0612  2                    [DS$K_BBS] :
;    0613  2                            Branch = .Branch_Value <0, 1, 0>;
;    0614  2
;    0615  2                    [DS$K_BBC] :
;    0616  2                            Branch = NOT .Branch_Value <0, 1, 0>;
```

```
;    0617  2                    !+
;    0618  2                    ! These are passed the value returned by the BUILTIN function for that branch (i.e.,
;    0619  2                    ! TESTBITSS for BBSS, etc.). Thus, the branch is based upon the value of the low bit.
;    0620  2                    !-
;    0621  2
;    0622  2                    [DS$K_BBSS] :
;    0623  2                            Branch = .Branch_Value <0, 1, 0>;
;    0624  2
;    0625  2                    [DS$K_BBCS] :
;    0626  2                            Branch = .Branch_Value <0, 1, 0>;
;    0627  2
;    0628  2                    [DS$K_BBSC] :
;    0629  2                            Branch = .Branch_Value <0, 1, 0>;
;    0630  2
;    0631  2                    [DS$K_BBCC] :
;    0632  2                            Branch = .Branch_Value <0, 1, 0>;
;    0633  2
;    0634  2                    [DS$K_BBSSI] :
;    0635  2                            Branch = .Branch_Value <0, 1, 0>;
;    0636  2
;    0637  2                    [DS$K_BBCCI] :
;    0638  2                            Branch = .Branch_Value <0, 1, 0>;
;    0639  2
;    0640  2                    !+
;    0641  2                    ! These are passed the value of the data whose low bit is being checked.
;    0642  2                    !-
;    0643  2
;    0644  2                    [DS$K_BLBS] :
;    0645  2                            Branch = .Branch_Value <0, 1, 0>;
;    0646  2
;    0647  2                    [DS$K_BLBC] :
;    0648  2                            Branch = NOT .Branch_Value <0, 1, 0>;
;    0649  2
;    0650  2                    !+
;    0651  2                    ! Treat the BERROR branch macro the same as the BNERROR branch macro. This means that the value  [05]
;    0652  2                    ! of R0 will remain the same for the two cases of QA not running and of QA running but not       [05]
;    0653  2                    ! forcing errors. When QA is running and this branch macro should be reversed-sensed (yuch),     [05]
;    0654  2                    ! then R0 will return from a BERROR with the low bit complemented (as will happen with a         [05]
;    0655  2                    ! BNERROR). However, upon return from this routine, the branch macro for BERROR will do a BLBC  [05]
;    0656  2                    ! instruction, as oposed to doing a BLBS (which is what the BNERROR macro will do). So, these   [05]
;    0657  2                    ! two branch macros will save the contents of R0 for the two cases above. They will also return [05]
;    0658  2                    ! the complemented low bit of R0 when forcing errors.                                          [05]
;    0659  2                    !-
;    0660  2
;    0661  2                    [DS$K_BERROR] :                         !                                                      [05]
;    0662  2                            Branch = .Branch_Value <0, 1, 0>;       ! These two should be the same.               [05]
;    0663  2
;    0664  2                    [DS$K_BNERROR] :                        !                                                      [05]
;    0665  2                            Branch = .Branch_Value <0, 1, 0>;       !                                             [05]
;    0666  2                    TES;
```

```
;    0667  2          !+
;    0668  2          ! If not running QA, simply return an indication of whether or not to branch.
;    0669  2          !-
;    0670  2
;    0671  2          IF (NOT .DSA$V_QA) THEN                                                          !
;    0672  3                BEGIN                                                                      !
;    0673  3                Save_Register_0 <0, 1, 0> = .Branch <0, 1, 0>;                             !
;    0674  4                RETURN (.Save_Register_0)                                                  !
;    0675  2                END;                                                                       !
;    0676  2
;    0677  2          !+
;    0678  2          ! Make the Branch value a one or a zero by clearing out all the other bits.
;    0679  2          !-
;    0680  2
;    0681  2          Branch = Boolean (.Branch);                                                      !
;    0682  2
;    0683  2          !+
;    0684  2          ! Save the error number and branch value. Call QA$Main with the appropriate branch code.
;    0685  2          !-
;    0686  2
;    0687  2          QA$W_Branch       = .Branch;
;    0688  2          QA$W_Error_Number = .Error_Number;
;    0689  2
;    0690  2          QA$MAIN (DSQA$K_QA_DSX$Branch);
;    0691  2
;    0692  2          !+
;    0693  2          ! Set the low bit of R0 to what was set up in the Error Phase One routine in the QA$W_Branch variable.
;    0694  2          ! If I am forcing errors, the low bit of R0 can change.
;    0695  2          !-
;    0696  2
;    0697  2          Save_Register_0 <0, 1, 0> = .QA$W_Branch <0, 1, 0>;                              !
;    0698  3          RETURN (.Save_Register_0)                                                        !
;    0699  1     END;


                                        .TITLE  QAMAIN *** QA Routines
                                        .IDENT  \1-08\

                                        .PSECT  WORK,NOEXE,2

00000 QA$AW_OVERALL_ERROR_TABLE:
                                        .BLKB   30
0001E QA$W_BRANCH::
                                        .BLKB   2
00020 QA$W_ERROR_NUMBER::
                                        .BLKB   2
00022 QA$W_SAVE_TEST::
                                        .BLKB   2
00024 QA$W_SAVE_LAST::
                                        .BLKB   2
00026                                   .BLKB   2
00028 QA$L_SAVED_DSA_FLAGS::
                                        .BLKB   4
0002C QA$AOB_ROUTINE_STATE::
                                        .BLKB   1
00020                                   .BLKB   3
00030 QA$AOB_CHECK_STATE::
```

```
                                                        .BLKB   2
                                            00032       .BLKB   2
                                            00034 QA$AOB_FLAGS::
                                                        .BLKB   1

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                 4E 49 41 4D 41 51 06   00000 P.AAA:  .ASCII  <6>\QAMAIN\                                 ;

                                                              DSA$AL_APTMAIL=     65024
                                                              DSA$GL_FLAGS=       65024
                                                              DSA$GL_APTCOM=      65028
                                                              DSA$GL_PASSES=      65032
                                                              DSA$GL_UNITS=       65036
                                                              DSA$GL_SECTNO=      65040
                                                              DSA$GL_SID=         65044
                                                              DSA$GL_MSGTYP=      65088
                                                              DSA$GL_ERRNO=       65092
                                                              DSA$GL_EVENT=       65096
                                                              DSA$GL_SUBTNO=      65100
                                                              DSA$GL_TESTNO=      65104
                                                              DSA$GL_PASSNO=      65108
                                                              DSA$GL_DEVLEN=      65112
                                                              DSA$GL_DEVNAM=      65116
                                                              DSA$GL_MSGPTR=      65128
                                                              $MODULE=            P.AAA
                                                        .EXTRN  QA$NORMAL_START
                                                        .EXTRN  QA$MULTIPLE_PASS
                                                        .EXTRN  QA$LOOP_ON_TEST
                                                        .EXTRN  QA$LOOP_ON_SUBTEST
                                                        .EXTRN  QA$RUN_BACKWARDS
                                                        .EXTRN  EXE$ALONONPAGED
                                                        .EXTRN  EXE$DEANONPAGED
                                                        .EXTRN  DSV$SHOWDEVICE, DSV$SHOWSELECT
                                                        .EXTRN  DSX$ABORT, VRSHOWFLG
                                                        .EXTRN  VRSHOWEF, DS$GL_FSTTEST
                                                        .EXTRN  DS$GL_LSTTEST, DS$GA_DS_CTRL_C_FIRST
                                                        .EXTRN  DS$GL_CLIBASE, DS$GB_WIDTH
                                                        .EXTRN  QA$W_TESTLOOPS, QA$W_SUBTESTLOOPS
                                                        .EXTRN  QA$W_MULTIPLEPASS
                                                        .EXTRN  QA$T_HEADER_1, QA$T_HEADER_2
                                                        .EXTRN  QA$T_HEADER_3, QA$T_OPERATOR_REQUEST_MESSAGE

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                  001E  00000       .ENTRY  DSX$BRANCH, Save R1,R2,R3,R4                    ; 0385
                                        54 00000000'  EF  9E  00002       MOVAB   QA$W_BRANCH, R4                         ;
                                           52         50  D0  00009       MOVL    REGISTER_0, SAVE_REGISTER_0             ; 0434
                                     27    00    04   AC  CF  0000C       CASEL   BRANCH_CODE, #0, #39                    ; 0436
                  0066       005D       0059       0050      00011 1$:   .WORD   2$-1$,-                                 ;
                  0078       006F       0066       0066      00019               3$-1$,-                                 ;
                  0085       0085       0085       007C      00021               4$-1$,-                                 ;
                  00A0       0098       0101       0090      00029               5$-1$,-                                 ;
                  00F6       00BA       00B2       00B2      00031               5$-1$,-                                 ;
                  00DE       00DE       00CA       00C2      00039               5$-1$,-                                 ;
                  00F6       0101       00EE       00E6      00041               6$-1$,-                                 ;
                  0101       0101       00F6       0101      00049               7$-1$,-                                 ;
```

```
                0101            0101            0101            0101    00051
                0101            0101            00F6            0101    00059

                                                                        8$-1$,-
                                                                        9$-1$,-
                                                                        9$-1$,-
                                                                        9$-1$,-
                                                                        11$-1$,-
                                                                        27$-1$,-
                                                                        12$-1$,-
                                                                        13$-1$,-
                                                                        15$-1$,-
                                                                        15$-1$,-
                                                                        17$-1$,-
                                                                        25$-1$,-
                                                                        18$-1$,-
                                                                        19$-1$,-
                                                                        21$-1$,-
                                                                        21$-1$,-
                                                                        22$-1$,-
                                                                        24$-1$,-
                                                                        27$-1$,-
                                                                        25$-1$,-
                                                                        27$-1$,-
                                                                        25$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$,-
                                                                        27$-1$
                51  D4  00061 2$:       CLRL    R1                      ;0443
0C              AC  D5  00063           TSTL    BRANCH_VALUE
                61  18  00066           BGEQ    16$
                33  11  00068           BRB     10$
                51  D4  0006A 3$:       CLRL    R1                      ;0446
                5B  11  0006C           BRB     16$
                51  D4  0006E 4$:       CLRL    R1                      ;0449
0C              AC  D5  00070           TSTL    BRANCH_VALUE
                54  14  00073           BGTR    16$
                26  11  00075           BRB     10$
                51  D4  00077 5$:       CLRL    R1                      ;0452
0C              AC  D5  00079           TSTL    BRANCH_VALUE
                7F  12  0007C           BNEQ    23$
                1D  11  0007E           BRB     10$
                51  D4  00080 6$:       CLRL    R1                      ;0461
0C              AC  D5  00082           TSTL    BRANCH_VALUE
                76  19  00085           BLSS    23$
                14  11  00087           BRB     10$
                51  D4  00089 7$:       CLRL    R1                      ;0464
                10  11  0008B           BRB     10$
                51  D4  0008D 8$:       CLRL    R1                      ;0467
0C              AC  D5  0008F           TSTL    BRANCH_VALUE
                69  15  00092           BLEQ    23$
                07  11  00094           BRB     10$
                51  D4  00096 9$:       CLRL    R1                      ;0476
```

```
                                                 0C   AC   D5 00098          TSTL    BRANCH_VALUE                                    ;
                                                      7B   13 0009B          BEQL    28$                                             ;
                                                      51   D6 0009D 10$:      INCL    R1                                              ;
                                                      77   11 0009F          BRB     28$                                             ;
        51       0C    AC            01               03   EF 000A1 11$:      EXTZV   #3, #1, BRANCH_VALUE, R1                        ; 0487
                                                      6F   11 000A7          BRB     28$                                             ;
        51       0C    AC            01               03   EF 000A9 12$:      EXTZV   #3, #1, BRANCH_VALUE, R1                        ; 0504
                                                      06   11 000AF          BRB     14$                                             ;
        51       0C    AC            01               00   EF 000B1 13$:      EXTZV   #0, #1, BRANCH_VALUE, R1                        ; 0513
        53       0C    AC            01               02   EF 000B7 14$:      EXTZV   #2, #1, BRANCH_VALUE, R3                        ;
                       50            51               53   89 000BD          BISB3   R3, R1, BRANCH                                  ;
                                                      58   11 000C1          BRB     29$                                             ; 0508
        51       0C    AC            01               02   EF 000C3 15$:      EXTZV   #2, #1, BRANCH_VALUE, R1                        ; 0528
                                                      4D   11 000C9 16$:      BRB     28$                                             ;
        51       0C    AC            01               03   EF 000CB 17$:      EXTZV   #3, #1, BRANCH_VALUE, R1                        ; 0536
                                                      3A   11 000D1          BRB     26$                                             ;
        51       0C    AC            01               03   EF 000D3 18$:      EXTZV   #3, #1, BRANCH_VALUE, R1                        ; 0553
                                                      06   11 000D9          BRB     20$                                             ;
        51       0C    AC            01               00   EF 000DB 19$:      EXTZV   #0, #1, BRANCH_VALUE, R1                        ; 0562
        53       0C    AC            01               02   EF 000E1 20$:      EXTZV   #2, #1, BRANCH_VALUE, R3                        ;
                                     53               53   D2 000E7          MCOML   R3, R3                                          ;
                                     51               53   CA 000EA          BICL2   R3, R1                                          ;
                                                      1E   11 000ED          BRB     26$                                             ; 0561
        51       0C    AC            01               02   EF 000EF 21$:      EXTZV   #2, #1, BRANCH_VALUE, R1                        ; 0577
                                                      16   11 000F5          BRB     26$                                             ;
        51       0C    AC            01               01   EF 000F7 22$:      EXTZV   #1, #1, BRANCH_VALUE, R1                        ; 0585
                                                      19   11 000FD 23$:      BRB     28$                                             ;
        51       0C    AC            01               01   EF 000FF 24$:      EXTZV   #1, #1, BRANCH_VALUE, R1                        ; 0593
                                                      06   11 00105          BRB     26$                                             ;
        51       0C    AC            01               00   EF 00107 25$:      EXTZV   #0, #1, BRANCH_VALUE, R1                        ; 0648
                                     50               51   92 0010D 26$:      MCOMB   R1, BRANCH                                      ;
                                                      09   11 00110          BRB     29$                                             ;
        51       0C    AC            01               00   EF 00112 27$:      EXTZV   #0, #1, BRANCH_VALUE, R1                        ; 0665
                                     50               51   90 00118 28$:      MOVB    R1, BRANCH                                      ;
                                          0000FE01    9F   95 0011B 29$:      TSTB    @#^X0000FE01                                    ; 0671
                                                      07   19 00121          BLSS    30$                                             ;
        52            01               00               50   F0 00123          INSV    BRANCH, #0, #1, SAVE_REGISTER_0                 ; 0673
                                                      1E   11 00128          BRB     31$                                             ; 0674
        51            50               01               00   EF 0012A 30$:      EXTZV   #0, #1, BRANCH, R1                              ; 0681
                                       50               51   90 0012F          MOVB    R1, BRANCH                                      ;
                                       64               50   9B 00132          MOVZBW  BRANCH, QA$W_BRANCH                             ; 0687
                                02     A4        08     AC   B0 00135          MOVW    ERROR_NUMBER, QA$W_ERROR_NUMBER                 ; 0688
                                                      11   DD 0013A          PUSHL   #17                                             ; 0690
                       00000000V      EF               01   FB 0013C          CALLS   #1, QA$MAIN                                     ;
        52            01               00               64   F0 00143          INSV    QA$W_BRANCH, #0, #1, SAVE_REGISTER_0            ; 0697
                                       50               52   D0 00148 31$:      MOVL    SAVE_REGISTER_0, R0                             ; 0698
                                                      04   00148          RET                                                     ; 0699

; Routine Size:  332 bytes,    Routine Base:  CODE + 0000
```

```
;    0700  1      %SBTTL 'QA$Abort_QA'
;    0701  1
;    0702  1      ROUTINE QA$Abort_QA : NOVALUE =                          !                    This whole routine is [02]
;    0703  1
;    0704  1      !++
;    0705  1      ! FUNCTIONAL DESCRIPTION:
;    0706  1      !
;    0707  1      !       Dump out debugging information, print the overall error summary table, restore the DSA flags,
;    0708  1      !       and abort the diagnostic (which will cause DS_CLEANUP to be executed, then branches to BEGIN).
;    0709  1      !
;    0710  1      ! CALLERS:
;    0711  1      !
;    0712  1      !       QA$Main
;    0713  1      !
;    0714  1      ! FORMAL PARAMETERS:
;    0715  1      !
;    0716  1      !       NONE
;    0717  1      !
;    0718  1      ! IMPLICIT INPUTS:
;    0719  1      !
;    0720  1      !       DSA$V_QA            - The QA bit in the DSA flags. It is cleared to indicate QA is no longer running.
;    0721  1      !       QA$L_Saved_QA_Flags - The saved DSA flags. The flags are restored here to what they were before
;    0722  1      !                             the START/QA (or RUN/QA) command started executing.
;    0723  1      !
;    0724  1      ! IMPLICIT OUTPUTS:
;    0725  1      !
;    0726  1      !       Debugging information, Overall Error Summary Table, and an abortion message.
;    0727  1      !
;    0728  1      ! COMPLETION CODES:
;    0729  1      !
;    0730  1      !       NONE - This routine does not return to its caller(s). It calls the DSX$Abort routine, which
;    0731  1      !              branches to BEGIN, which goes back to DS$Cli.
;    0732  1      !
;    0733  1      ! SIDE EFFECTS:
;    0734  1      !
;    0735  1      !       The diagnostic is aborted.
;    0736  1      !
;    0737  1      !--
```

```
;   0738  2     BEGIN                                                   !                                                       [02]
;   0739  2             QA$Dump ();                                     ! Dump out information for the user                     [02]
;   0740  2             QA$Print_Overall_Errors ();                     ! Print the summary table                               [02]
;   0741  2             DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;           ! Restore DSA flags                                     [02]
;   0742  2             QA$Init ();                                     ! Initialize for next run                               [02]
;   0743  2             DSA$V_QA = Off;                                 ! Turn off QA                                           [02]
;   0744  2             DSX$Abort ()                                    ! Abort the diagnostic because of the error             [02]
;   0745  2                                                             ! that was found. DSX$Abort will not return to here!    [02]
;   0746  1     END;

                                        0000 00000 QA$ABORT_QA:
                                                   .WORD   Save nothing                                                         ; 0702
                00000000V  EF             00  FB 00002              CALLS   #0, QA$DUMP                                         ; 0739
                00000000V  EF             00  FB 00009              CALLS   #0, QA$PRINT_OVERALL_ERRORS                         ; 0740
                0000FE00   9F 00000000'   EF  D0 00010              MOVL    QA$L_SAVED_DSA_FLAGS, @#^X0000FE00                  ; 0741
                00000000V  EF             00  FB 0001B              CALLS   #0, QA$INIT                                         ; 0742
                0000FE01   9F         80  8F  8A 00022              BICB2   #128, @#^X0000FE01                                  ; 0743
                00000000G  EF             00  FB 0002A              CALLS   #0, DSX$ABORT                                       ; 0744
                                          04 00031                  RET                                                         ; 0746

; Routine Size:  50 bytes,    Routine Base:  CODE + 014C
```

```
;    0747  1      %SBTTL 'QA$Add_Forced_Error'
;    0748  1
;    0749  1      GLOBAL ROUTINE QA$Add_Forced_Error (Error_Number : WORD) : NOVALUE =
;    0750  1
;    0751  1      !++
;    0752  1      ! FUNCTIONAL DESCRIPTION:
;    0753  1      !
;    0754  1      !       This routine adds an error to the Forced Error Summary Table.
;    0755  1      !
;    0756  1      ! CALLER (S):
;    0757  1      !
;    0758  1      !       The Error check routines.
;    0759  1      !
;    0760  1      ! FORMAL PARAMETERS:
;    0761  1      !
;    0762  1      !       Error_Number : The DS error number.
;    0763  1      !
;    0764  1      ! IMPLICIT INPUTS:
;    0765  1      !
;    0766  1      !       QA$A_Error_List_Head - Pointer to the head of a linked list of Forced-Error data entries.
;    0767  1      !       QA$A_Prev_Test_Tail  - Pointer to the previous test's last entry in the linked list.
;    0768  1      !
;    0769  1      ! IMPLICIT OUTPUTS:
;    0770  1      !
;    0771  1      !       NONE
;    0772  1      !
;    0773  1      ! COMPLETION CODES:
;    0774  1      !
;    0775  1      !       NONE
;    0776  1      !
;    0777  1      ! SIDE EFFECTS:
;    0778  1      !
;    0779  1      !       Adds a record to the linked list of forced-error records.
;    0780  1      !
;    0781  1      ! FIGURES:
;    0782  1      !
;    0783  1      !       The Forced-Error Record:
;    0784  1      !
;    0785  1      !       ------------------------------------------------------------------------------------------
;    0786  1      !       |Error_Number| Test_Number|      Error_Count      |      Subtest_Number   |      Next_Error       |
;    0787  1      !       |            |            |                       |                       |                       |
;    0788  1      !       ------------------------------------------------------------------------------------------
;    0789  1      !
;    0790  1      !       Error_Number   - The error number passed to the $DS_ERRxxxx routine.
;    0791  1      !       Test_Number    - The test that the error occurred in.
;    0792  1      !       Error_Count    - The number of times that error has been forced.
;    0793  1      !       Subtest_Number - The subtest that the error occurred in.
;    0794  1      !       Next_Error     - The pointer to the next Forced-Error record in the linked list, or nil.
;    0795  1      !--
```

```
;    0796  1     %SBTTL 'New'
;    0797  2     BEGIN
;    0798  2     !++
;    0799  2     ! FUNCTIONAL DESCRIPTION:
;    0800  2     !
;    0801  2     !       This routine returns a 16-byte vector for use by the QA$Add_Forced_Error routine.
;    0802  2     !
;    0803  2     ! CALLER (S):
;    0804  2     !
;    0805  2     !       QA$K_Add_Forced_Error
;    0806  2     !
;    0807  2     ! FORMAL PARAMETERS:
;    0808  2     !
;    0809  2     !       Record_Ptr - Contains the address of the 16-byte vector.
;    0810  2     !
;    0811  2     ! IMPLICIT INPUTS:
;    0812  2     !
;    0813  2     !       NONE
;    0814  2     !
;    0815  2     ! IMPLICIT OUTPUTS:
;    0816  2     !
;    0817  2     !       NONE
;    0818  2     !
;    0819  2     ! COMPLETION CODES:
;    0820  2     !
;    0821  2     !       NONE
;    0822  2     !
;    0823  2     ! SIDE EFFECTS:
;    0824  2     !
;    0825  2     !       Will abort the QA sequence if the returned size of the vector is not 16, or if the vector
;    0826  2     !       cannot be allocated.
;    0827  2     !
;    0828  2     !--
```

```
:    0829  2      !
:    0830  2      !       ROUTINE New (Record_Ptr) : NOVALUE =
:    0831  2      !       BEGIN
:    0832  2      !               LOCAL
:    0833  2      !                       Returned_Size;
:    0834  2      !
:    0835  2      !               IF (NOT EXE$AloNonPaged (QA$K_Forced_Error_Record_Size; Returned_Size, .Record_Ptr)) THEN
:    0836  2      !                       BEGIN
:    0837  2      !                       Logic_Error ('Cannot allocate forced-error record');
:    0838  2      !                       QA$Abort_QA ();
:    0839  2      !                       END;
:    0840  2      !
:    0841  2      !               IF (.Returned_Size NEQ QA$K_Forced_Error_Record_Size) THEN
:    0842  2      !                       BEGIN
:    0843  2      !                       Logic_Error ('Wrong size forced-error record allocated');
:    0844  2      !                       QA$Abort_QA ();
:    0845  2      !                       END;
:    0846  2      !       END;
```

```
;    0847  2     !
;    0848  2     !SBTTL 'Add_Error_Record'
;    0849  2     !
;    0850  2     !       ROUTINE Add_Error_Record (Record_Ptr, Error_Number : WORD) : NOVALUE =
;    0851  2     !
;    0852  2     !       !++
;    0853  2     !       ! FUNCTIONAL DESCRIPTION:
;    0854  2     !       !
;    0855  2     !       !       This routine adds one record to the linked list of forced-error records. It also fills in the
;    0856  2     !       !       fields of the new record. The Error_Count is assumed to be one. The Test and Subtest numbers
;    0857  2     !       !       are taken from the DSA area. This assumes that the forced-error is being added as they occur
;    0858  2     !       !       (i.e., there is no change in the test/subtest numbers from the time the error is forced to the
;    0859  2     !       !       time it is added).
;    0860  2     !       !
;    0861  2     !       ! CALLER (S):
;    0862  2     !       !
;    0863  2     !       !       QA$Add_Forced_Error
;    0864  2     !       !
;    0865  2     !       ! FORMAL PARAMETERS:
;    0866  2     !       !
;    0867  2     !       !       Record_Ptr      - Contains the address of the record which will preceed this new record.
;    0868  2     !       !       Error_Number    - The error number that will be stored in the new record.
;    0869  2     !       !
;    0870  2     !       ! IMPLICIT INPUTS:
;    0871  2     !       !
;    0872  2     !       !       DSA$GL_TESTNO - The DSA test number.
;    0873  2     !       !       DSA$GL_SUBTNO - The DSA subtest number.
;    0874  2     !       !
;    0875  2     !       ! IMPLICIT OUTPUTS:
;    0876  2     !       !
;    0877  2     !       !       NONE
;    0878  2     !       !
;    0879  2     !       ! COMPLETION CODES:
;    0880  2     !       !
;    0881  2     !       !       NONE
;    0882  2     !       !
;    0883  2     !       ! SIDE EFFECTS:
;    0884  2     !       !
;    0885  2     !       !       NONE
;    0886  2     !       !
;    0887  2     !       !--
```

```
;    0888  2      !
;    0889  2      !       BEGIN
;    0890  2      !               LOCAL
;    0891  2      !                       New_Record_Ptr;
;    0892  2      !
;    0893  2      !               MAP
;    0894  2      !                       DSA$GL_TESTNO : WORD,
;    0895  2      !                       Record_Ptr    : Error_Record;
;    0896  2      !
;    0897  2      !               New (New_Record_Ptr);
;    0898  2      !
;    0899  2      !               BEGIN
;    0900  2      !                       MAP
;    0901  2      !                               New_Record_Ptr : Error_Record;
;    0902  2      !
;    0903  2      !                       New_Record_Ptr [QA$VW_Error_Number]   = .Error_Number;
;    0904  2      !                       New_Record_Ptr [QA$VW_Test_Number]    = .DSA$GL_TESTNO;
;    0905  2      !                       New_Record_Ptr [QA$VL_Subtest_Number] = .DSA$GL_SUBTNO;
;    0906  2      !                       New_Record_Ptr [QA$VL_Error_Count]    = 1;
;    0907  2      !                       New_Record_Ptr [QA$VA_Next_Error]     = .Record_Ptr [QA$VA_Next_Error]
;    0908  2      !               END;
;    0909  2      !
;    0910  2      !               Record_Ptr [QA$VA_Next_Error] = .New_Record_Ptr;
;    0911  2      !       END;
```

```
;    0912  2      !
;    0913  2      %SBTTL 'QA$Add_Forced_Error'
;    0914  2
;    0915  2              LOCAL
;    0916  2                      Test : WORD,                            ! The DSA$GL_TESTNO.
;    0917  2
;    0918  2                      Subtest,                                ! The DSA$GL_SUBTNO.
;    0919  2
;    0920  2                      Prev_Record_Ptr,                        ! The pointer to the record before the Record_Ptr record.
;    0921  2
;    0922  2                      Record_Ptr;                             ! The pointer used to sneak through the linked list.
;    0923  2
;    0924  2              !+
;    0925  2              ! Check to see if the Error_List_Head is nil. If so, this is the first record to be added. Add a new
;    0926  2              ! record that contains all zero fields. Then, add another record after this one that will contain the
;    0927  2              ! correct information. The null record is used to make adding the first record easier. It also helps
;    0928  2              ! facilitate adding records before the first record. Set both of the global linked list pointers to
;    0929  2              ! point to this null record.
;    0930  2              !-
;    0931  2
;    0932  2              IF (.QA$A_Error_List_Head EQL QA$K_Nil) THEN
;    0933  2                      BEGIN
;    0934  2                      LOCAL
;    0935  2                              New_Record_Ptr;
;    0936  2
;    0937  2                      New (New_Record_Ptr);
;    0938  2
;    0939  2                      BEGIN
;    0940  2                              MAP
;    0941  2                                      New_Record_Ptr : Error_Record;
;    0942  2
;    0943  2                              New_Record_Ptr [QA$VW_Error_Number]   = 0;
;    0944  2                              New_Record_Ptr [QA$VW_Test_Number]    = 0;
;    0945  2                              New_Record_Ptr [QA$VL_Subtest_Number] = 0;
;    0946  2                              New_Record_Ptr [QA$VL_Error_Count]    = 0;
;    0947  2                              New_Record_Ptr [QA$VA_Next_Error]     = QA$K_Nil;
;    0948  2                      END;
;    0949  2
;    0950  2                      Add_Error_Record (.New_Record_Ptr, .Error_Number);
;    0951  2
;    0952  2                      QA$A_Prev_Test_Tail  = .New_Record_Ptr;
;    0953  2                      QA$A_Error_List_Head = .New_Record_Ptr;
;    0954  2                      RETURN
;    0955  2                      END;
```

```
;    0956  2       !
;    0957  2       !+
;    0958  2       ! Start looking at the current head of the linked list that has all the test and subtest numbers
;    0959  2       ! the same. This list appears at the end of the forced-error linked list.
;    0960  2       !-
;    0961  2       !
;    0962  2       BEGIN
;    0963  2               MAP
;    0964  2                       QA$A_Prev_Test_Tail : Error_Record;
;    0965  2       !
;    0966  2               Prev_Record_Ptr = .QA$A_Prev_Test_Tail;
;    0967  2               Record_Ptr      = .QA$A_Prev_Test_Tail [QA$VA_Next_Error]
;    0968  2       END;
;    0969  2       !
;    0970  2       BEGIN
;    0971  2               MAP
;    0972  2                       Record_Ptr : Error_Record;
;    0973  2       !
;    0974  2               Test    = .Record_Ptr [QA$VW_Test_Number];
;    0975  2               Subtest = .Record_Ptr [QA$VL_Subtest_Number]
;    0976  2       END;
;    0977  2       !
;    0978  2       !+
;    0979  2       ! Check the start of the last link of like test/subtest numbers. See if this forced error happened
;    0980  2       ! in a new test and/or subtest. If so, add one record to the end of this list.
;    0981  2       !-
;    0982  2       !
;    0983  2       IF ((.Test NEQ .DSA$GL_TESTNO) OR (.Subtest NEQ .DSA$GL_SUBTNO)) THEN
;    0984  2               BEGIN
;    0985  2               !+
;    0986  2               ! Then we have forced an error call in a new test and/or subtest. Find the end of the linked list
;    0987  2               ! of forced-errors and add a new record onto the end. Save a pointer to the previous end record.
;    0988  2               !-
;    0989  2               MAP
;    0990  2                       Record_Ptr : Error_Record;
;    0991  2               !
;    0992  2               While (.Record_Ptr [QA$VA_Next_Error] NEQ QA$K_Nil) DO
;    0993  2                       Record_Ptr = .Record_Ptr [QA$VA_Next_Error];
;    0994  2               !
;    0995  2               QA$A_Prev_Test_Tail = .Record_Ptr;
;    0996  2               !
;    0997  2               Add_Error_Record (.Record_Ptr, .Error_Number);
;    0998  2               RETURN
;    0999  2               END;
```

```
1000  2      !
1001  2      !+
1002  2      ! Thus, at this point we do NOT have:
1003  2      !       1. A new list (i.e., no records).
1004  2      !       2. A new test and/or subtest number (i.e., the error number being added happened in the same
1005  2      !          test and subtest as the previously-added error number happened in.
1006  2      ! So, look along the error numbers in the linked list of records whose test/subtest numbers match for the
1007  2      ! correct place to insert this error number.
1008  2      !-
1009  2
1010  2      BEGIN
1011  2              MAP
1012  2                      Record_Ptr : Error_Record;
1013  2
1014  2              LOCAL
1015  2                      Curr_Error_Number;
1016  2
1017  2              WHILE (1) DO
1018  2                      BEGIN
1019  2                      Curr_Error_Number = .Record_Ptr [QA$VW_Error_Number];
1020  2
1021  2                      IF (.Curr_Error_Number LSS .Error_Number) THEN
1022  2                              BEGIN
1023  2                              !+
1024  2                              ! If the next one is nil, add the new one at the end. If it is not nil,
1025  2                              ! advance the pointer and continue looping.
1026  2                              !-
1027  2                              IF (.Record_Ptr [QA$VA_Next_Error] EQL QA$K_Nil) THEN
1028  2                                      BEGIN
1029  2                                      Add_Error_Record (.Record_Ptr, .Error_Number);
1030  2                                      RETURN
1031  2                                      END
1032  2                              ELSE
1033  2                                      BEGIN
1034  2                                      Prev_Record_Ptr = .Record_Ptr;
1035  2                                      Record_Ptr      = .Record_Ptr [QA$VA_Next_Error]
1036  2                                      END
1037  2                              END
1038  2                      ELSE IF (.Curr_Error_Number EQL .Error_Number) THEN
1039  2                              BEGIN
1040  2                              !+
1041  2                              ! This error number is already in the linked list. Add one to the total number
1042  2                              ! of times it has occurred.
1043  2                              !-
1044  2                              Record_Ptr [QA$VL_Error_Count] = .Record_Ptr [QA$VL_Error_Count] + 1;
1045  2                              RETURN
1046  2                              END
1047  2                      ELSE                    ! Curr_Error_Number GTR Error_Number
1048  2                              BEGIN
1049  2                              !+
1050  2                              ! This error number belongs just before the current record being looked at.
1051  2                              ! Therefore, add a new record there.
1052  2                              !-
1053  2                              Add_Error_Record (.Prev_Record_Ptr, .Error_Number);
1054  2                              RETURN
1055  2                              END;
1056  2      END;                    ! WHILE ...
```

```
;   1057  2     !       END;                                    ! BEGIN ...
;   1058  2
;   1059  2             1;                                      !
;   1060  2
;   1061  1     END;



                                                0000 00000          .ENTRY  QA$ADD_FORCED_ERROR, Save nothing           ; 0749
                                                  04 00002          RET                                                 ; 1061

; Routine Size:  3 bytes,    Routine Base:  CODE + 017E
```

```
;    1062  1     %SBTTL 'QA$Add_QA_Error'
;    1063  1
;    1064  1     GLOBAL ROUTINE QA$Add_QA_Error (Routine_Constant) : NOVALUE =
;    1065  1
;    1066  1     !++
;    1067  1     ! FUNCTIONAL DESCRIPTION:
;    1068  1     !
;    1069  1     !       This routine adds an error to the QA$AW_Overall_Error_Table word vector.
;    1070  1     !
;    1071  1     ! CALLER(S):
;    1072  1     !
;    1073  1     !       The check routines.
;    1074  1     !
;    1075  1     ! FORMAL PARAMETERS:
;    1076  1     !
;    1077  1     !       Routine_Constant - One of the QA$K_check-name constants.
;    1078  1     !
;    1079  1     ! IMPLICIT INPUTS:
;    1080  1     !
;    1081  1     !       QA$AW_Overall_Error_Table
;    1082  1     !
;    1083  1     ! IMPLICIT OUTPUTS:
;    1084  1     !
;    1085  1     !       NONE
;    1086  1     !
;    1087  1     ! COMPLETION CODES:
;    1088  1     !
;    1089  1     !       NONE
;    1090  1     !
;    1091  1     ! SIDE EFFECTS:
;    1092  1     !
;    1093  1     !       Adds one to the Routine_Constant entry in the QA$AW_Overall_Error_Table.
;    1094  1     !
;    1095  1     !--
;    1096  1
;    1097  2     BEGIN
;    1098  2             QA$AW_Overall_Error_Table [.Routine_Constant] =
;    1099  2                     .QA$AW_Overall_Error_Table [.Routine_Constant] + 1
;    1100  1     END;


                                        0000 00000          .ENTRY  QA$ADD_QA_ERROR, Save nothing                   ; 1064
                                50      04   AC   D0 00002  MOVL    ROUTINE_CONSTANT, R0                            ; 1098
                                   00000000'EF40   B6 00006  INCW    QA$AW_OVERALL_ERROR_TABLE[R0]                   ; 1099
                                        04 0000D            RET                                                     ; 1100

; Routine Size:  14 bytes,    Routine Base:  CODE + 0181
```

```
;    1101  1      %SBTTL  'QA$Control_C_Handler'
;    1102  1
;    1103  1      ROUTINE QA$Control_C_Handler =                          !                                  [06]
;    1104  1
;    1105  1      !++                                                                                        [06]
;    1106  1      ! FUNCTIONAL DESCRIPTION:                                                                  [06]
;    1107  1      !                                                                                          [06]
;    1108  1      !       This routine is called by the Supervisor's Control-C (i.e., ^C) handler, KB_Check in the KERNEL.MAR  [06]
;    1109  1      !       module. It will abort QA. This routine is set up as a handler by QA$Main.          [06]
;    1110  1      !                                                                                          [06]
;    1111  1      ! FORMAL PARAMETERS:                                                                       [06]
;    1112  1      !                                                                                          [06]
;    1113  1      !       NONE.                                                                              [06]
;    1114  1      !                                                                                          [06]
;    1115  1      ! CALLER(S):                                                                               [06]
;    1116  1      !                                                                                          [06]
;    1117  1      !       KB_CHECK in the KERNEL.MAR module.                                                 [06]
;    1118  1      !                                                                                          [06]
;    1119  1      ! IMPLICIT INPUTS:                                                                         [06]
;    1120  1      !                                                                                          [06]
;    1121  1      !       NONE                                                                               [06]
;    1122  1      !                                                                                          [06]
;    1123  1      ! IMPLICIT OUTPUTS:                                                                        [06]
;    1124  1      !                                                                                          [06]
;    1125  1      !       NONE                                                                               [06]
;    1126  1      !                                                                                          [06]
;    1127  1      ! COMPLETION CODES:                                                                        [06]
;    1128  1      !                                                                                          [06]
;    1129  1      !       None, really, it is defined as a VALUE routine however, because KB_CHECK looks at the return status  [06]
;    1130  1      !       by this routine (however, this routine does not return)!                           [06]
;    1131  1      !                                                                                          [06]
;    1132  1      ! SIDE EFFECTS:                                                                            [06]
;    1133  1      !                                                                                          [06]
;    1134  1      !       Aborts the diagnostic and the QA sequence of checks. There are no completion codes because control does  [06]
;    1135  1      !       not return to here after the DSX$Abort routine call. Control will eventually get to DS$Cli in the  [06]
;    1136  1      !       CLI.MAR module.                                                                    [06]
;    1137  1      !                                                                                          [06]
;    1138  1      !--                                                                                        [06]
```

```
;    1139  2     BEGIN                                                    !                                                [06]
;    1140  2             $DS_CNTRLC (DISABL=1);                           ! Disable Control-C's                            [06]
;    1141  2             DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;            ! Restore DSA flags                              [06]
;    1142  2             QA$Init ();                                      ! Initialize for next run                        [06]
;    1143  2             DSA$V_QA = Off;                                  ! Turn off QA                                    [06]
;    1144  2             DSX$Abort ();                                    ! Abort the diagnostic, return to CLI.           [06]
;    1145  3             RETURN (1)                                       ! This will never be executed, because of the    [06]
;    1146  3                                                              ! above Abort.                                   [06]
;    1147  1     END;                                                     !                                                [06]


                                                        .EXTRN  DS$CNTRLC

                                        0000 00000 QA$CONTROL_C_HANDLER:
                                                        .WORD   Save nothing                                    ; 1103
                                    01  DD 00002           PUSHL   #1                                          ; 1140
                                    7E  D4 00004           CLRL    -(SP)                                       ;
                00000000G     9F    02  FB 00006           CALLS   #2, @#DS$CNTRLC                             ;
                0000FE00      9F 00000000'  EF  D0 00000   MOVL    QA$L_SAVED_DSA_FLAGS, @#^X0000FE00          ; 1141
                00000000V     EF    00  FB 00018           CALLS   #0, QA$INIT                                 ; 1142
                0000FE01      9F      80  8F  8A 0001F     BICB2   #128, @#^X0000FE01                          ; 1143
                00000000G     EF    00  FB 00027           CALLS   #0, DSX$ABORT                               ; 1144
                              50    01  D0 0002E           MOVL    #1, R0                                      ; 1145
                                        04 00031           RET                                                 ; 1147

; Routine Size:  50 bytes,    Routine Base:  CODE + 018F
```

```
;    1148  1      %SBTTL 'QA$Dump'
;    1149  1
;    1150  1      ROUTINE QA$Dump : NOVALUE =
;    1151  1
;    1152  1      !++
;    1153  1      ! FUNCTIONAL DESCRIPTION:
;    1154  1      !
;    1155  1      !       This routine dumps pertinent information whenever a QA error is
;    1156  1      !       detected. If the users diagnostic has been aborted (i.e., an
;    1157  1      !       INIT_CONTEXT has been done), this routine will fail because the
;    1158  1      !       user's call frames are gone.
;    1159  1      !
;    1160  1      ! CALLER:
;    1161  1      !
;    1162  1      !       QA$Main - This routine should be called when a QA error is
;    1163  1      !                 detected. QA$Main should be its only caller.
;    1164  1      !
;    1165  1      ! FORMAL PARAMETERS:
;    1166  1      !
;    1167  1      !       NONE
;    1168  1      !
;    1169  1      ! IMPLICIT INPUTS:
;    1170  1      !
;    1171  1      !       Various DSA and diagnostic header (i.e., L$x_) values and flags.
;    1172  1      !
;    1173  1      ! IMPLICIT OUTPUTS:
;    1174  1      !
;    1175  1      !       If possible, helpful debugging information is printed.
;    1176  1      !
;    1177  1      ! COMPLETION CODES:
;    1178  1      !
;    1179  1      !       NONE
;    1180  1      !
;    1181  1      ! SIDE EFFECTS:
;    1182  1      !
;    1183  1      !       NONE
;    1184  1      !
;    1185  1      !--
```

```
;    1186  2     BEGIN
;    1187  2             BUILTIN
;    1188  2                     FP, SP, AP,
;    1189  2                     MOVPSL;
;    1190  2
;    1191  2             LITERAL
;    1192  2                     Reg_Not_Saved      = %X'00000000',      ! Special value to indicate that the register wasn't saved
;    1193  2                     Max_Regs_Saved     = 12;                ! Maximum that can be saved in a call frame.
;    1194  2
;    1195  2             LOCAL
;    1196  2                     Frame_Start,                            ! Start of the user's call frame
;    1197  2
;    1198  2                     !+
;    1199  2                     ! Stuff contained in a call frame:
;    1200  2                     !-
;    1201  2
;    1202  2                     Diag_PSW : WORD,                        ! Only get PSW - not PSL
;    1203  2                     Register_Mask,                          ! Call frame register mask
;    1204  2                     Diag_AP,
;    1205  2                     Diag_FP,
;    1206  2                     Diag_PC,                                ! Saved registers
;    1207  2                     Diag_SP,
;    1208  2
;    1209  2                     Number_Of_Regs;                         ! Number of regs in the register save mask
;    1210  2
;    1211  2             LOCAL
;    1212  2                     Fake_CLI_Table : VECTOR [4, LONG],              ! Fake DS$GL_CLIBASE
;    1213  2                     Reg_Vector     : VECTOR [Max_Regs_Saved, LONG]; ! Store 12 registers
;    1214  2
;    1215  2             BIND
;    1216  2                     Dump_Header_4 = $ASCIC ('!25* QA ERROR DETECTED!/:/'),
;    1217  2                     Dump_Line_1   = $ASCIC ('Test: !3UL!9* Subtest: !3UL!6* Pass: !3UL!9* Section: !AC!/!/'),    !    [02]
;    1218  2                     Dump_Line_2   = $ASCIC ('R2:  !8XL(X)!2* R3:  !8XL(X)!2* R4:  !8XL(X)!2* R5:  !8XL(X)!/'),  !    [02]
;    1219  2                     Dump_Line_3   = $ASCIC ('R6.  !8XL(X)!2* R7:  !8XL(X)!2* R8:  !8XL(X)!2* R9:  !8XL(X)!/'),  !    [02]
;    1220  2                     Dump_Line_4   = $ASCIC ('R10: !8XL(X)!2* R11: !8XL(X)!2* AP:  !8XL(X)!2* FP:  !8XL(X)!/'),  !    [02]
;    1221  2                     Dump_Line_5   = $ASCIC ('SP:  !8XL(X)!2* PC:  !8XL(X)!2* PSW: !8XW(X)!/'),                  !    [07]
;    1222  2                     Dump_Line_7   = $ASCIC ('!/SHOW FLAGS:!/'),                                                 !    [07]
;    1223  2                     Dump_Line_8   = $ASCIC ('!/SHOW EVENT FLAGS:!/'),                                           !    [07]
;    1224  2                     Dump_Line_9   = $ASCIC ('!/SHOW DEVICE:!/'),                                                !    [07]
;    1225  2                     Dump_Line_10  = $ASCIC ('!/SHOW SELECTED:!/');                                              !    [07]
```

```
;  1226  2     !+
;  1227  2     ! Picture of a call frame:
;  1228  2     !
;  1229  2     !       ------------------------------------------------------------------------
;  1230  2     !       :                        PUSHx arguments for the CALLx                 : High Memory
;  1231  2     !       ------------------------------------------------------------------------
;  1232  2     !       :                                        : Number of arguments         : :(AP)
;  1233  2     !       ------------------------------------------------------------------------
;  1234  2     !       :                        Saved R11                                     : +n+4
;  1235  2     !       ------------------------------------------------------------------------
;  1236  2     !       :                        Saved R10                                     : +n
;  1237  2     !       ------------------------------------------------------------------------
;  1238  2     !                                       .
;  1239  2     !                                       .
;  1240  2     !                                       .
;  1241  2     !       ------------------------------------------------------------------------
;  1242  2     !       :                        Saved R0                                      : +20
;  1243  2     !       ------------------------------------------------------------------------
;  1244  2     !       :                        Saved PC                                      : +16
;  1245  2     !       ------------------------------------------------------------------------
;  1246  2     !       :                        Saved FP                                      : +12
;  1247  2     !       ------------------------------------------------------------------------
;  1248  2     !       :                        Saved AP                                      : +8
;  1249  2     !       ------------------------------------------------------------------------
;  1250  2     !       : SPA : S : 0 :Register save mask <11:0>:        PSW <15:5>        : 0  : +4
;  1251  2     !       ------------------------------------------------------------------------
;  1252  2     !       :                        Condition Handler (Initially Zero)            : :(FP) - Low Memory
;  1253  2     !       ------------------------------------------------------------------------
;  1254  2     !       :                        Start of called routine's stack area          :
;  1255  2     !       ------------------------------------------------------------------------
;  1256  2     !-
```

```
;    1257  2              !+
;    1258  2              ! Print the header lines for this routine.
;    1259  2              !-
;    1260  2
;    1261  2              $DS_PRINTF (QA$T_Header_1);                                  !                                                  [01]
;    1262  2              $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);   !                                                  [01]
;    1263  2              $DS_PRINTF (QA$T_Header_3, 0);                               ! Specify 0 to get system time.                    [01]
;    1264  2              $DS_PRINTF (Dump_Header_4);
;    1265  2
;    1266  2              !+
;    1267  2              ! Print the test number, subtest number, pass number, and section name.
;    1268  2              !-
;    1269  2
;    1270  3              BEGIN
;    1271  3                      BIND
;    1272  3                              Sec_Name_Table = ((.L$A_SECNAM) + 4) : VECTOR [, LONG];
;    1273  3
;P   1274  3                      $DS_PRINTF (Dump_Line_1, .DSA$GL_TESTNO,
;P   1275  3                                  .DSA$GL_SUBTNO, .DSA$GL_PASSNO,
;    1276  4                                  .Sec_Name_Table [.DSA$GL_SECTNO])
;    1277  2              END;
;    1278  2
;    1279  2              !+
;    1280  2              ! If an Init_Context has been done, then cannot follow stack frames back to the user's last
;    1281  2              ! call frame. Therefore, just print above lines and return.
;    1282  2              !-
;    1283  2
;    1284  2              IF (.QA$A0B_Flags [QA$K_Init_Context_Done]) THEN
;    1285  3                      BEGIN
;    1286  3                      QA$A0B_Flags [QA$K_Init_Context_Done] = False;  ! Turn it back off
;    1287  3                      RETURN
;    1288  2                      END;
```

```
;    1289  2              !+
;    1290  2              ! If cannot follow stack frames back to the user's program, the Find_User_Call_Frame routine will
;    1291  2              ! print an error message.
;    1292  2              !-
;    1293  2
;    1294  2              IF (NOT QA$Find_User_Call_Frame (Frame_Start)) THEN
;    1295  3                      BEGIN
;    1296  3                      RETURN                                  ! Failure
;    1297  2                      END;
;    1298  2
;    1299  2              !+
;    1300  2              ! Frame_Start now equals the start of the user's last call frame. Set Diag_FP equal to the start of
;    1301  2              ! the frame that was just found. This frame is the one from which I am extracting everything, so it
;    1302  2              ! makes sense to use the FP that points to this frame. This is done instead of using the FP that is
;    1303  2              ! contained in the frame we are looking at (i.e., that points to the frame of the caller of the
;    1304  2              ! user's routine that bombed).
;    1305  2              !-
;    1306  2
;    1307  3              BEGIN
;    1308  3                      BIND
;    1309  3                              Call_Frame = .Frame_Start : BLOCK [, BYTE];
;    1310  3
;    1311  3                      Diag_PSW        = .Call_Frame [SF$W_SAVE_PSW];
;    1312  3                      Register_Mask   = .Call_Frame [SF$V_SAVE_MASK];
;    1313  3                      Diag_AP         = .Call_Frame [SF$L_SAVE_AP];
;    1314  3                      Diag_FP         = .Frame_Start;
;    1315  3                      Diag_PC         = .Call_Frame [SF$L_SAVE_PC];
;    1316  2              END;
```

```
;    1317  2              !+
;    1318  2              ! Get those registers that were saved. Assume Reg_Not_Saved for non-saved ones. This will only get
;    1319  2              ! those registers that were saved in the user's last call frame. That is, those registers saved by
;    1320  2              ! the last called Supervisor routine, which through a series of CALL's, BSBW's, etc. got to here.
;    1321  2              ! However, those Supervisor routines which, by virtue of their being called by the diagnostic,      [02]
;    1322  2              ! cause a QA error must have all the registers (i.e., R2 through R11) saved in the entry mask for   [02]
;    1323  2              ! their routine. This allows the following code to access all these registers.                      [02]
;    1324  2              !-
;    1325  2
;    1326  2              INCR Reg_Number FROM 0 TO (Max_Regs_Saved - 1) DO
;    1327  2                      Reg_Vector [.Reg_Number] = Reg_Not_Saved;
;    1328  2
;    1329  2              !+
;    1330  2              ! Number_Of_Regs will actually end up to be one less than the actual number of registers that were
;    1331  2              ! saved. This is so that I can use it to index the Start_Registers vector, which starts at zero, not one!
;    1332  2              !-
;    1333  2
;    1334  2              Number_Of_Regs = -1;
;    1335  2
;    1336  2              INCR Bit_Number FROM 0 TO (Max_Regs_Saved - 1) DO
;    1337  3                      BEGIN
;    1338  3                      BIND
;    1339  3                              !+
;    1340  3                              ! Start_registers is the start of a vector of saved registers in the call frame. This
;    1341  3                              ! vector starts at offset 20 from the start of the frame.
;    1342  3                              !-
;    1343  3
;    1344  3                              Start_Registers = ((.Frame_Start) + 20) : VECTOR [, LONG];
;    1345  3
;    1346  3                      MAP
;    1347  3                              Register_Mask    : BITVECTOR [32];       ! Reference as a Bit Vector
;    1348  3
;    1349  3                      IF (.Register_Mask [.Bit_Number]) THEN
;    1350  4                              BEGIN
;    1351  4
;    1352  4                              Number_Of_Regs = .Number_Of_Pegs + 1;
;    1353  4                              Reg_Vector [.Bit_Number] = .Start_Registers [.Number_Of_Regs];
;    1354  3                              END;
;    1355  2              END;
```

```
;  1356  2              Number_Of_Regs = .Number_Of_Regs + 1;
;  1357  2
;  1358  2              !+
;  1359  2              ! Number_Of_Regs now equals the actual number of registers that were saved in the call frame.
;  1360  2              ! Compute the value of the Stack Pointer before the user's CALLx was executed. This is found
;  1361  2              ! by computing the size of the last call frame put on by the user. This size is the added to
;  1362  2              ! the starting address of the call frame and the stack pointer alignment is added in,
;  1363  2              ! producing the value of the SP.
;  1364  2              !-
;  1365  2
;  1366  3              BEGIN
;  1367  3                      BIND
;  1368  3                              Call_Frame = .Frame_Start : BLOCK [, BYTE];
;  1369  3
;  1370  3                      Diag_SP = .Frame_Start +                        ! Start of call frame
;  1371  3                                (5 * 4) +                             ! 5 longwords
;  1372  3                                (.Number_Of_Regs * 4) +               ! Longword for each reg
;  1373  3                                .Call_Frame [SF$V_STACKOFFS];         ! SPA
;  1374  2              END;
;  1375  2
;  1376  2              !+
;  1377  2              ! Print the registers and the PSW. Do not print R0 and R1 since they may not be correct.          [02]
;  1378  2              !-
;  1379  2
;  1380  2              $DS_PRINTF (Dump_line_2, .Reg_Vector [2],  .Reg_Vector [3],  .Reg_Vector [4],  .Reg_Vector [5]);  !  [02]
;  1381  2              $DS_PRINTF (Dump_line_3, .Reg_Vector [6],  .Reg_Vector [7],  .Reg_Vector [8],  .Reg_Vector [9]);  !  [02]
;  1382  2              $DS_PRINTF (Dump_line_4, .Reg_Vector [10], .Reg_Vector [11], .Diag_AP,         .Diag_FP);       !  [02]
;  1383  2              $DS_PRINTF (Dump_line_5, .Diag_SP,         .Diag_PC,         .Diag_PSW);                       !  [02]
```

```
;    1384  2              !+
;    1385  2              ! Print the control flags, event flags, and selected devices.
;    1386  2              !-
;    1387  2
;    1388  2              $DS_PRINTF (Dump_line_7);
;    1389  2              VRShowFlg ();                                   ! Show the control flags
;    1390  2
;    1391  2              $DS_PRINTF (Dump_line_8);
;    1392  2              VRShowEf  ();                                   ! Show the event flags
;    1393  2
;    1394  2              !+
;    1395  2              ! Set up a fake CLI database table. This is needed because the following two routines access the
;    1396  2              ! CLI$Q_FILE quadword in this table in order to determine what device should be SHOW'n. All that
;    1397  2              ! is needed in this table is a null quadword at the CLI$Q_FILE offset. Zero out the entire table
;    1398  2              ! to play it safe though. Only 4 longwords are allocated since the FILE quadword starts at the
;    1399  2              ! third longword in the real CLI table.
;    1400  2              !-
;    1401  2
;    1402  2              INCR I FROM 0 TO 3 DO
;    1403  2                      Fake_CLI_Table [.I] = 0;
;    1404  2
;    1405  2              $DS_PRINTF (Dump_line_9);
;    1406  2              DSV$ShowDevice (Fake_CLI_Table);                ! Show the devices
;    1407  2
;    1408  2              $DS_PRINTF (Dump_line_10);
;    1409  2              DSV$ShowSelect (Fake_CLI_Table)                 ! Show the selected devices
;    1410  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20 52 4F 52 52 45 20 41 51 20 2A 35 32 21 1A  00007 P.AAB:   .ASCII  <26>\!25* QA ERROR DETECTED!/!/\                    ;
   2F 21 2F 21 44 45 54 43 45 54 45 44         00016
20 2A 39 21 4C 55 33 21 20 3A 74 73 65 54 3D  00022 P.AAC:   .ASCII  \=Test: !3UL!9* Subtest: !3UL!6* Pass: !3\          ;
36 21 4C 55 33 21 20 3A 74 73 65 74 62 75 53  00031
   33 21 20 3A 73 73 61 50 20 2A              00040
20 3A 6E 6F 69 74 63 65 53 20 2A 39 21 4C 55  0004A          .ASCII  \UL!9* Section: !AC!/!/\                        ;
   2F 21 2F 21 43 41 21                       00059
32 21 29 58 28 4C 58 38 21 20 20 3A 32 52 3E  00060 P.AAD:   .ASCII  \>R2:  !8XL(X)!2* R3:  !8XL(X)!2* R4:  !8\          ;
21 29 58 28 4C 58 38 21 20 20 3A 33 52 20 2A  0006F
   38 21 20 20 3A 34 52 20 2A 32              0007E
21 20 20 3A 35 52 20 2A 32 21 29 58 28 4C 58  00088          .ASCII  \XL(X)!2* R5:  !8XL(X)!/\                       ;
   2F 21 29 58 28 4C 58 38                    00097
32 21 29 58 28 4C 58 38 21 20 20 3A 36 52 3E  0009F P.AAE:   .ASCII  \>R6:  !8XL(X)!2* R7:  !8XL(X)!2* R8:  !8\          ;
21 29 58 28 4C 58 38 21 20 20 3A 37 52 20 2A  000AE
   38 21 20 20 3A 38 52 20 2A 32              000BD
21 20 20 3A 39 52 20 2A 32 21 29 58 28 4C 58  000C7          .ASCII  \XL(X)!2* R9   !8XL(X)!/\                        ;
   2F 21 29 58 28 4C 58 38                    000D6
32 21 29 58 28 4C 58 38 21 20 3A 30 31 52 3E  000DE P.AAF:   .ASCII  \>R10: !8XL(X)!2* R11: !8XL(X)!2* AP:  !8\          ;
21 29 58 28 4C 58 38 21 20 3A 31 31 52 20 2A  000ED
   38 21 20 20 3A 50 41 20 2A 32              000FC
21 20 20 3A 50 46 20 2A 32 21 29 58 28 4C 58  00106          .ASCII  \XL(X)!2* FP:  !8XL(X)!/\                       ;
   2F 21 29 58 28 4C 58 38                    00115
32 21 29 58 28 4C 58 38 21 20 20 3A 50 53 2E  0011D P.AAG:   .ASCII  \.SP:  !8XL(X)!2* PC:  !8XL(X)!2* PSW: !8\          ;
21 29 58 28 4C 58 38 21 20 20 3A 43 50 20 2A  0012C
   38 21 20 3A 57 53 50 20 2A 32              0013B
```

```
                                                  2F 21 29 58 28 57 58  00145                .ASCII  \XW(X)!/\                                  ;
21 3A 53 47 41 4C 46 20 57 4F 48 53 2F 21 0F  00014C P.AAH:   .ASCII  <15>\!/SHOW FLAGS:!/\                                  ;
                                              2F  00158                                                                       ;
46 20 54 4E 45 56 45 20 57 4F 48 53 2F 21 15  00015C P.AAI:   .ASCII  <21>\!/SHOW EVENT FLAGS:!/\                              ;
                                              2F  00168                                                                       ;
3A 45 43 49 56 45 44 20 57 4F 48 53 2F 21 10  00172 P.AAJ:   .ASCII  <16>\!/SHOW DEVICE:!/\                                   ;
                                              2F  00181                                                                       ;
45 54 43 45 4C 45 53 20 57 4F 48 53 2F 21 12  00183 P.AAK:   .ASCII  <18>\!/SHOW SELECTED:!/\                                 ;
                                              2F  00192                                                                       ;

                                                                    DUMP_HEADER_4=      P.AAB
                                                                    DUMP_LINE_1=        P.AAC
                                                                    DUMP_LINE_2=        P.AAD
                                                                    DUMP_LINE_3=        P.AAE
                                                                    DUMP_LINE_4=        P.AAF
                                                                    DUMP_LINE_5=        P.AAG
                                                                    DUMP_LINE_7=        P.AAH
                                                                    DUMP_LINE_8=        P.AAI
                                                                    DUMP_LINE_9=        P.AAJ
                                                                    DUMP_LINE_10=       P.AAK
                                                                            .EXTRN  DS$PRINTF

                                                                            .PSECT  CODE,NOWRT,  SHR,2

                                                  0FFC 00000 QA$DUMP::.WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1150
                                              5B 00000000'  EF   9E 00002           MOVAB   QA$A0B_FLAGS, R11                 ;
                                              5A 00000000'  EF   9E 00009           MOVAB   DUMP_HEADER_4, R10                ;
                                              59 00000000G  9F   9E 00010           MOVAB   @#DS$PRINTF, R9                   ;
                                              5E        BC  AE   9E 00017           MOVAB   -68(SP), SP                       ;
                                                 00000000G  EF   9F 0001B           PUSHAB  QA$T_HEADER_1                     ; 1261
                                                 69         01   FB 00021           CALLS   #1, DS$PRINTF                     ;
                                                 7E 0000020C   9F   7D 00024           MOVQ    @#^X0000020C, -(SP)               ; 1262
                                                    00000208   9F   DD 0002B           PUSHL   @#^X00000208                      ;
                                                    00000000G  EF   9F 00031           PUSHAB  QA$T_HEADER_2                     ;
                                                 69         04   FB 00037           CALLS   #4, DS$PRINTF                     ;
                                                            7E   D4 0003A           CLRL    -(SP)                             ; 1263
                                                    00000000G  EF   9F 0003C           PUSHAB  QA$T_HEADER_3                     ;
                                                 69         02   FB 00042           CALLS   #2, DS$PRINTF                     ;
                                                            5A   DD 00045           PUSHL   R10                               ; 1264
                                                 69         01   FB 00047           CALLS   #1, DS$PRINTF                     ;
                                           51 00000250   9F   04   C1 0004A           ADDL3   #4, @#^X00000250, R1              ; 1272
                                                 50 0000FE10   9F   D0 00052           MOVL    @#^X0000FE10, R0                  ; 1276
                                                       6140   DD   00059           PUSHL   (R1)[R0]                          ;
                                                    0000FE54   9F   DD 0005C           PUSHL   @#^X0000FE54                      ;
                                                    0000FE4C   9F   DD 00062           PUSHL   @#^X0000FE4C                      ;
                                                    0000FE50   9F   DD 00068           PUSHL   @#^X0000FE50                      ;
                                                          1B   AA   9F 0006E           PUSHAB  DUMP_LINE_1                       ;
                                                 69         05   FB 00071           CALLS   #5, DS$PRINTF                     ;
                                           04    68         03   E1 00074           BBC     #3, QA$A0B_FLAGS, 1$              ; 1284
                                                 68         08   8A 00078           BICB2   #8, QA$A0B_FLAGS                  ; 1286
                                                            04   00078           RET                                       ; 1285
                                                            5E   DD 0007C 1$:      PUSHL   SP                                ; 1294
                                              00000000V     EF   01   FB 0007E           CALLS   #1, QA$FIND_USER_CALL_FRAME       ;
                                                 01         50   E8 00085           BLBS    R0, 2$                            ;
                                                            04   00088           RET                                       ;
                                                 51         6E   D0 00089 2$:      MOVL    FRAME_START, R1                   ; 1309
                                                 58        04   A1   B0 0008C           MOVW    4(R1), DIAG_PSW                   ; 1311
```

```
                54      06    A1          0C      00    EF 00090          EXTZV   #0, #12, 6(R1), REGISTER_MASK          ; 1312
                                          57  08  A1    D0 00096          MOVL    8(R1), DIAG_AP                         ; 1313
                                          56      51    D0 0009A          MOVL    R1, DIAG_FP                            ; 1314
                                          55  10  A1    D0 0009D          MOVL    16(R1), DIAG_PC                        ; 1315
                                              50  D4    000A1            CLRL    REG_NUMBER                             ; 1326
                                          04 AE40 D4    000A3 3$:        CLRL    REG_VECTOR[REG_NUMBER]                 ; 1327
                        F8                50      0B    F3 000A7         AOBLEQ  #11, REG_NUMBER, 3$
                                          50      01    CE 000AB         MNEGL   #1, NUMBER_OF_REGS                     ; 1334
                                          53  14  A1    9E 000AE         MOVAB   20(R1), R3                             ; 1344
                                              52  D4    000B2            CLRL    BIT_NUMBER
                        08                54      52    E1 000B4 4$:     BBC     BIT_NUMBER, REGISTER_MASK, 5$          ; 1349
                                              50  D6    000B8            INCL    NUMBER_OF_REGS                         ; 1352
                                    04 AE42       6340  D0 000BA         MOVL    (R3)[NUMBER_OF_REGS], REG_VECTOR-      ; 1353
                                                                                  [BIT_NUMBER]
                        F0                52      0B    F3 000C0 5$:     AOBLEQ  #11, BIT_NUMBER, 4$                    ; 1336
                                              50  D6    000C4            INCL    NUMBER_OF_REGS                         ; 1356
                                          50      6140  DE 000C6         MOVAL   (R1)[NUMBER_OF_REGS], R0               ; 1371
                51      07    A1          02      06    EF 000CA         EXTZV   #6, #2, 7(R1), R1                      ; 1373
                                          52  14  A140  9E 000D0         MOVAB   20(R1)[R0], DIAG_SP                    ; 1372
                                              18  AE    DD 000D5         PUSHL   REG_VECTOR+20                          ; 1380
                                              18  AE    DD 000D8         PUSHL   REG_VECTOR+16
                                              18  AE    DD 000DB         PUSHL   REG_VECTOR+12
                                              18  AE    DD 000DE         PUSHL   REG_VECTOR+8
                                              59  AA    9F 000E1         PUSHAB  DUMP_LINE_2
                                          69          05  FB 000E4        CALLS   #5, DS$PRINTF
                                              28  AE    DD 000E7         PUSHL   REG_VECTOR+36                          ; 1381
                                              28  AE    DD 000EA         PUSHL   REG_VECTOR+32
                                              28  AE    DD 000ED         PUSHL   REG_VECTOR+28
                                              28  AE    DD 000F0         PUSHL   REG_VECTOR+24
                                            0098  CA    9F 000F3         PUSHAB  DUMP_LINE_3
                                          69          05  FB 000F7        CALLS   #5, DS$PRINTF
                                              56  DD    000FA            PUSHL   DIAG_FP                                ; 1382
                                              57  DD    000FC            PUSHL   DIAG_AP
                                              38  AE    DD 000FE         PUSHL   REG_VECTOR+44
                                              38  AE    DD 00101         PUSHL   REG_VECTOR+40
                                            00D7  CA    9F 00104         PUSHAB  DUMP_LINE_4
                                          69          05  FB 00108        CALLS   #5, DS$PRINTF
                                          7E          58  3C 0010B         MOVZWL  DIAG_PSW, -(SP)                        ; 1383
                                                      24  BB 0010E         PUSHR   #^M<R2,R5>
                                            0116  CA    9F 00110         PUSHAB  DUMP_LINE_5
                                          69          04  FB 00114        CALLS   #4, DS$PRINTF
                                            0145  CA    9F 00117         PUSHAB  DUMP_LINE_7                            ; 1388
                                          69          01  FB 0011B        CALLS   #1, DS$PRINTF
                                    00000000G         EF  16 0011E         JSB     VRSHOWFLG                              ; 1389
                                            0155  CA    9F 00124         PUSHAB  DUMP_LINE_8                            ; 1391
                                          69          01  FB 00128        CALLS   #1, DS$PRINTF
                                    00000000G         EF  16 0012B         JSB     VRSHOWEF                               ; 1392
                                                      50  D4 00131         CLRL    I                                      ; 1402
                                              34  AE40  D4 00133 6$:       CLRL    FAKE_CLI_TABLE[I]                      ; 1403
                        F8                50      03    F3 00137         AOBLEQ  #3, I, 6$
                                            016B  CA    9F 0013B         PUSHAB  DUMP_LINE_9                            ; 1405
                                          69          01  FB 0013F        CALLS   #1, DS$PRINTF
                                          52          34  AE  9E 00142     MOVAB   FAKE_CLI_TABLE, R2                     ; 1406
                                    00000000G         EF  16 00146         JSB     DSV$SHOWDEVICE
                                            017C  CA    9F 0014C         PUSHAB  DUMP_LINE_10                           ; 1408
                                          69          01  FB 00150        CALLS   #1, DS$PRINTF
                                          52          34  AE  9E 00153     MOVAB   FAKE_CLI_TABLE, R2                     ; 1409
```

```
                                00000000G  EF  16 00157         JSB     DSV$SHOWSELECT                          ;
                                           04 0015D         RET                                                 ; 1410

; Routine Size:  350 bytes,    Routine Base:  CODE + 01C1
```

```
;    1411  1      %SBTTL  'QA$Find_User_Call_Frame'
;    1412  1
;    1413  1      GLOBAL ROUTINE QA$Find_User_Call_Frame (Frame_Start) =
;    1414  1
;    1415  1      !++
;    1416  1      ! FUNCTIONAL DESCRIPTION:
;    1417  1      !
;    1418  1      !       Find the user's last call frame by following the linked list
;    1419  1      !       of call frames. Find the first call frame whose saved PC is
;    1420  1      !       less than the starting location of the Supervisor. This should
;    1421  1      !       insure that that call frame was put on the stack when the user
;    1422  1      !       called a Supervisor service routine. This called frame will
;    1423  1      !       contain the saved PC, AP, FP, PSW, perhaps some of the general
;    1424  1      !       registers, and (indirectly) the SP.
;    1425  1      !
;    1426  1      ! CALLER:
;    1427  1      !
;    1428  1      !       QA$Dump - to find the call frame so it can print the debugging
;    1429  1      !                 information.
;    1430  1      !
;    1431  1      ! FORMAL PARAMETERS:
;    1432  1      !
;    1433  1      !       Frame_Start - will contain the starting location of the
;    1434  1      !                     found call frame.
;    1435  1      !
;    1436  1      ! IMPLICIT INPUTS:
;    1437  1      !
;    1438  1      !       NONE
;    1439  1      !
;    1440  1      ! IMPLICIT OUTPUTS:
;    1441  1      !
;    1442  1      !       Messages indicating that the call frame list cannot be followed.
;    1443  1      !
;    1444  1      ! COMPLETION CODES:
;    1445  1      !
;    1446  1      !       R0 EQL 1 => Success, Frame_Start contains the starting location
;    1447  1      !                                of the call frame.
;    1448  1      !       R0 EQL 1 => Failure, Frame_Start contains crap.
;    1449  1      !
;    1450  1      ! SIDE EFFECTS:
;    1451  1      !
;    1452  1      !       Hopefully none.
;    1453  1      !
;    1454  1      !--
```

```
;    1455  2      BEGIN
;    1456  2              BIND
;    1457  2                      FP_Ptr = .Frame_Start;
;    1458  2
;    1459  2              BUILTIN
;    1460  2                      FP, PROBEW;
;    1461  2
;    1462  2              LITERAL
;    1463  2                      Max_Frames_Checked = 25;                ! Maximum number of frames to check
;    1464  2
;    1465  2              LOCAL
;    1466  2                      Last_PC,
;    1467  2                      Number_Frames_Checked;
```

B 1 SCAN$SEARCHLIST Search list fo
C 1 SCAN$DEVICE Scan a device name
D 1 SCAN$DEVICE Scan a device name
E 1 Breakup File name into parts
F 1 Breakup File name into parts
G 1 Symbol table
H 1 Symbol table
I 1 Cross reference
J 1 Cross reference
K 1 Cross reference
L 1 Cross reference
M 1 Cross reference
N 1 Cross reference
B 2 PCSLOAD Module
C 2 PCSLOAD Module
D 2 Libraries
E 2 Table of Contents
F 2 Psect Definitions
G 2 PCSLOAD-Static Storage
H 2 DSX$LOADPCS Routine
I 2 DSX$LOADPCS Routine
J 2 DSV$INITPCS Routine
K 2 DSV$INITPCS Routine
L 2 DSV$INITPCS Routine
M 2 *** PRINT Formatted print rout
N 2 *** PRINT Formatted print rout
B 3 *** PRINT Formatted print rout
C 3 *** PRINT Formatted print rout
D 3 Libraries, Macros, and Equated
E 3 Work Psect Declarations
F 3 Data Psect Declarations
G 3 DSV$SetPage Routine - Set term
H 3 DSV$SetPage Routine - Set term
I 3 DSV$ShowPage Routine - Show te
J 3 DSV$ShowPage Routine - Show te
K 3 DSX$Type_Out Routine - Do actu
L 3 DSX$Type_Out Routine - Do actu
M 3 DSX$Type_Out Routine - Do actu
N 3 DSX$Type_Out Routine - Do actu
B 4 DSX$Type_Out Routine - Do actu
C 4 DSX$Print Routine - Load type
D 4 DSX$Print Routine - Load type
E 4 DSX$PrintX Routines - Extended
F 4 DSX$PrintX Routines - Extended
G 4 DSX$PrintX Routines - Extended
H 4 DSX$PrintX Routines - Extended
I 4 DSX$CvtReg Routine - Mnemonic
J 4 DSX$CvtReg Routine - Mnemonic
K 4 DSX$CvtReg Routine - Mnemonic
L 4 DSX$CvtReg Routine - Mnemonic
M 4 DSX$CvtReg Routine - Mnemonic
N 4 DSX$CvtReg Routine - Mnemonic
B 5 Symbol table
C 5 Symbol table
D 5 Symbol table
E 5 Psect synopsis
F 5 Cross reference
G 5 Cross reference
H 5 Cross reference
I 5 Cross reference
J 5 Cross reference
K 5 Cross reference
L 5 Cross reference
M 5 Cross reference
N 5 *** PROBE Check address access
B 6 *** PROBE Check address access
C 6 DSX$PROBE
D 6 DSX$PROBE
E 6 DSX$PROBE
F 6 ACC$HANDLER
G 6 ACC$HANDLER
H 6 ACC$HANDLER
I 6 - UDA50 BOOT DRIVER
J 6 - UDA50 BOOT DRIVER
K 6 DECLARATIONS
L 6 DECLARATIONS
M 6 DECLARATIONS
N 6 UDA50 Bootstrap device initial
B 7 UDA50 Bootstrap device initial
C 7 UDA50 Bootstrap device initial
D 7 UDA50 Bootstrap driver QIO
E 7 UDA50 Bootstrap driver QIO
F 7 Symbol table
G 7 Psect synopsis
H 7 Cross reference
I 7 Cross reference
J 7 Cross reference
K 7 Cross reference
L 7 *** QA Check Routines
M 7 Table of Contents
N 7 Libraries
B 8 Included Macros and Literals
C 8 External Routines
D 8 External Own Storage
E 8 Psect Definitions
F 8 Local Macro Definitions
G 8 Local Macro Definitions
H 8 Module-Global Static Storage
I 8 ASCIC Bindings
J 8 ASCIC Bindings
K 8 ASCIC Bindings
L 8 QA$Normal_Start
M 8 QA$Normal_Start
N 8 QA$Normal_Start
B 9 QA$Normal_Start
C 9 QA$Normal_Start
D 9 QA$Normal_Start
E 9 QA$Normal_Start
F 9 QA$Multiple_Pass
G 9 QA$Multiple_Pass
H 9 QA$Multiple_Pass
I 9 QA$Multiple_Pass
J 9 QA$Loop_On_Test
K 9 QA$Loop_On_Test
L 9 QA$Loop_On_Test
M 9 QA$Loop_On_Test
N 9 QA$Loop_On_Test
B 10 QA$Loop_On_Test
C 10 QA$Loop_On_Subtest
D 10 QA$Loop_On_Subtest
E 10 QA$Loop_On_Subtest
F 10 QA$Loop_On_Subtest
G 10 QA$Loop_On_Subtest
H 10 QA$Loop_On_Subtest
I 10 QA$Loop_On_Subtest
J 10 QA$Run_Backwards
K 10 QA$Run_Backwards
L 10 QA$Run_Backwards
M 10 QA$Run_Backwards
N 10 QA$Run_Backwards
B 11 QA$Run_Backwards
C 11 QA$Run_Backwards
D 11 QA$Run_Backwards
E 11 QA$Run_Backwards
F 11 QA$Run_Backwards
G 11 QA$Run_Backwards
H 11 QA$Run_Backwards
I 11 End of Module QACHECKS
J 11 *** Setting and Showing QA Fla
K 11 *** Setting and Showing QA Fla
L 11 Table of Contents
M 11 Psect Definitions
N 11 Literals
B 12 Static Storage
C 12 Global Bindings
D 12 VRSetQA
E 12 VRSetQA
F 12 VRSetQA
G 12 VRSetQA
H 12 VRShowQA
I 12 VRShowQA
J 12 VRShowQA
K 12 VRShowQA
L 12 End of Module QAFLAGS
M 12 *** QA Routines
N 12 *** QA Routines
B 13 Linkage Definitions
C 13 Table of Contents
D 13 Libraries
E 13 Included Macros and Literals
F 13 External Routines
G 13 External Own Storage
H 13 Psect Definitions
I 13 Local Macro Definitions
J 13 Local Macro Definitions
K 13 Module-Global Literals
L 13 Field Definitions
M 13 Module-Global Static Storage
N 13 Supervisor-Global Static Stora
B 14 DSX$BRANCH
C 14 DSX$BRANCH
D 14 DSX$BRANCH
E 14 DSX$BRANCH
F 14 DSX$BRANCH
G 14 DSX$BRANCH
H 14 DSX$BRANCH
I 14 DSX$BRANCH
J 14 DSX$BRANCH
K 14 DSX$BRANCH
L 14 QA$Abort_QA

```
M 14  QA$Abort_QA
N 14  QA$Add_Forced_Error
B 15  New
C 15  New
D 15  New
E 15  New
F 15  New
G 15  New
H 15  New
I 15  New
J 15  QA$Add_QA_Error
K 15  QA$Control_C_Handler
L 15  QA$Control_C_Handler
M 15  QA$Dump
N 15  QA$Dump
B 16  QA$Dump
C 16  QA$Dump
D 16  QA$Dump
E 16  QA$Dump
F 16  QA$Dump
G 16  QA$Dump
H 16  QA$Dump
I 16  QA$Dump
J 16  QA$Dump
K 16  QA$Find_User_Call_Frame
L 16  QA$Find_User_Call_Frame
```

```
;    1468  2              !+
;    1469  2              ! Find the last call frame put on the stack by the user. Verify that the FP in the call frame
;    1470  2              ! can be accessed. This partially insures that I am not following invalid call frames.
;    1471  2              !-
;    1472  2
;    1473  2              FP_Ptr = .FP;                                   ! Start at the current FP
;    1474  2              Number_Frames_Checked = 0;
;    1475  2
;    1476  2              WHILE 1 DO
;    1477  3                      BEGIN
;    1478  3
;    1479  3                      IF (NOT PROBEW (%REF (0), %REF (3), .FP_Ptr)) THEN
;    1480  4                              BEGIN
;    1481  4                              BIND
;  P 1482  4                                      Invalid_Call_Frame = $ASCIC
;    1483  4                                              ('!/**QA Internal Error: Invalid call frame. FP = !XL(X).!/!/'); !      [01]
;    1484  4
;    1485  4                              $DS_PRINTF (Invalid_Call_Frame, .FP_Ptr);
;    1486  5                              RETURN (QA$K_Failure)
;    1487  3                              END;
;    1488  3
;    1489  3                      IF ((Number_Frames_Checked = .Number_Frames_Checked + 1) GTR Max_Frames_Checked) THEN
;    1490  4                              BEGIN
;    1491  4                              BIND
;  P 1492  4                                      Too_Many_Frames = $ASCIC
;    1493  4                              ('!/**QA Internal Error: Cannot follow stack frames back to the diagnostic!/!/'); !      [01]
;    1494  4
;    1495  4                              $DS_PRINTF (Too_Many_Frames);
;    1496  5                              RETURN (QA$K_Failure)
;    1497  3                              END;
;    1498  3
;    1499  3                      !+
;    1500  3                      ! The contents of FP_Ptr is the starting address of the current frame being looked at. Pick out the
;    1501  3                      ! PC from the call frame. If this PC is less than the starting location of the Supervisor, then this
;    1502  3                      ! call frame must be the last call frame of the user. If not, backtrack to the next call frame.
;    1503  3                      !-
;    1504  3
;    1505  3                      Last_PC = .(.FP_Ptr + 16);                      ! PC is offset 16 from the start of the call frame.
;    1506  3
;    1507  3                      IF (.Last_PC LSS DS$K_DS_Starting_Location) THEN
;    1508  4                              BEGIN
;    1509  5                              RETURN (QA$K_Success)
;    1510  3                              END;
;    1511  3
;    1512  3                      FP_Ptr = .(.FP_Ptr + 12);                       ! FP is offset 12 from the start of the call frame.
;    1513  3                      END
;    1514  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  3B  00196 P.AAL:  .ASCII  \;!/**QA Internal Error: Invalid call fra\  ;
69  6C  61  76  6E  49  20  3A  72  6F  72  72  45  20  6C  001A5                                                            ;
                    61  72  66  20  6C  6C  61  63  20  64  001B4                                                            ;
29  58  28  4C  58  21  20  3D  20  50  46  20  2E  65  6D  001BE          .ASCII  \me. FP = !XL(X).!/!/\                    ;
                                    2F  21  2F  21  2E  001CD                                                            ;
```

```
61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  4C   001D2 P.AAM:  .ASCII  \L!/**QA Internal Error: Cannot follow st\  ;
74  6F  6E  6E  61  43  20  3A  72  6F  72  72  45  20  6C   001E1                                                            ;
                    74  73  20  77  6F  6C  6C  6F  66  20   001F0                                                            ;
6B  63  61  62  20  73  65  6D  61  72  66  20  6B  63  61   001FA          .ASCII  \ack frames back to the diagnostic!/!/\      ;
73  6F  6E  67  61  69  64  20  65  6E  74  20  6F  74  20   00209                                                            ;
                            2F  21  2F  21  33  69  74       00218                                                            ;

                                                                    INVALID_CALL_FRAME= P.AAL
                                                                    TOO_MANY_FRAMES=    P.AAM


                                                                    .PSECT  CODE,NOWRT,  SHR,2

                                                      003C 00000            .ENTRY  QA$FIND_USER_CALL_FRAME, Save R2,R3,R4,R5   ; 1413
                                55 00000000G    9F    9E 00002            MOVAB   @#DS$PRINTF, R5                             ;
                             04 BC              5D    D0 00009            MOVL    FP, @FRAME_START                            ; 1473
                                                53    D4 0000D            CLRL    NUMBER_FRAMES_CHECKED                       ; 1474
                                52        04    BC    D0 0000F 1$:        MOVL    @FRAME_START, R2                            ; 1479
                          62                    00    0D 00013            PROBEW  #0, #3, (R2)                                ;
                                03              0D    12 00017            BNEQ    2$                                          ;
                                                52    DD 00019            PUSHL   R2                                          ; 1485
                                00000000'       EF    9F 0001B            PUSHAB  INVALID_CALL_FRAME                          ;
                                65              02    FB 00021            CALLS   #2, DS$PRINTF                               ;
                                                10    11 00024            BRB     3$                                          ; 1486
                                                53    D6 00026 2$:        INCL    NUMBER_FRAMES_CHECKED                       ; 1489
                                19              53    D1 00028            CMPL    NUMBER_FRAMES_CHECKED, #25                  ;
                                                0C    15 0002B            BLEQ    4$                                          ;
                                00000000'       EF    9F 0002D            PUSHAB  TOO_MANY_FRAMES                             ; 1495
                                65              01    FB 00033            CALLS   #1, DS$PRINTF                               ;
                                                50    D4 00036 3$:        CLRL    R0                                          ; 1496
                                                      04 00038            RET                                                 ;
                                54        10    A2    D0 00039 4$:        MOVL    16(R2), LAST_PC                             ; 1505
                       00010000 8F              54    D1 0003D            CMPL    LAST_PC, #65536                             ; 1507
                                                04    18 00044            BGEQ    5$                                          ;
                                50              01    D0 00046            MOVL    #1, R0                                      ; 1509
                                                      04 00049            RET                                                 ;
                             04 BC        0C    A2    D0 0004A 5$:        MOVL    12(R2), @FRAME_START                        ; 1512
                                                BE    11 0004F            BRB     1$                                          ; 1476

; Routine Size:  81 bytes,    Routine Base:  CODE + 031F
```

```
;    1515  1      %SBTTL 'QA$Init'
;    1516  1
;    1517  1      ROUTINE QA$Init : NOVALUE =
;    1518  1
;    1519  1      !++
;    1520  1      ! FUNCTIONAL DESCRIPTION:
;    1521  1      !
;    1522  1      !       This routine performs all the global initializations required
;    1523  1      !       for the QA routines. This routine must be called both when a
;    1524  1      !       START or RUN command is being processed and when a QA execution
;    1525  1      !       completes (to set up a subsequent execution).
;    1526  1      !
;    1527  1      ! CALLER(S):
;    1528  1      !
;    1529  1      !       QA$Main
;    1530  1      !
;    1531  1      ! FORMAL PARAMETERS:
;    1532  1      !
;    1533  1      !       NONE
;    1534  1      !
;    1535  1      ! IMPLICIT INPUTS:
;    1536  1      !
;    1537  1      !       The QA$AOB_Check_State bit vector.
;    1538  1      !       The QA$AW_Overall_Error_Table word vector.
;    1539  1      !       The QA$AOB_Flags bit vector.
;    1540  1      !
;    1541  1      ! IMPLICIT OUTPUTS:
;    1542  1      !
;    1543  1      !       The QA$AOB_Check_State bit vector.
;    1544  1      !       The QA$AW_Overall_Error_Table word vector.
;    1545  1      !       The QA$AOB_Flags bit vector.
;    1546  1      !
;    1547  1      ! COMPLETION CODES:
;    1548  1      !
;    1549  1      !       NONE
;    1550  1      !
;    1551  1      ! SIDE EFFECTS:
;    1552  1      !
;    1553  1      !       The QA$AOB_Check_State bit vector is zeroed.
;    1554  1      !       The QA$AW_Overall_Error_Table word vector is zeroed.
;    1555  1      !       The QA$AOB_Flags bit vector is zeroed.
;    1556  1      !--
;    1557  1
;    1558  2      BEGIN
;    1559  2              INCR Flag FROM 0 TO (QA$K_Number_QA_Flags - 1) DO
;    1560  2                      QA$AOB_Flags [.Flag] = Off;
;    1561  2
;    1562  2              INCR Bit_Pos FROM 0 TO (QA$K_Number_Of_Checks - 1) DO
;    1563  2                      QA$AOB_Check_State [.Bit_Pos] = Off;
;    1564  2
;    1565  2              INCR Error_Entry FROM 0 TO (QA$K_Number_Of_Checks - 1) DO
;    1566  2                      QA$AW_Overall_Error_Table [.Error_Entry] = 0;
;    1567  2
;    1568  2      !       QA$A_Prev_Test_Tail  = QA$K_Nil;
;    1569  2      !       QA$A_Error_List_Head = QA$K_Nil;
;    1570  2              DS$GA_DS_Ctrl_C_First = 0;       ! Disable Supervisor catching of Control-C's.          [06]
;    1571  1      END;
```

```
                                                0004 00000 QA$INIT:.WORD   Save R2                                          ; 1517
                                  52 00000000'   EF   9E 00002          MOVAB   QA$AOB_FLAGS, R2                          ;
                                                 50   D4 00009          CLRL    FLAG                                      ; 1559
                          00                     50   E5 0000B 1$:      BBCC    FLAG, QA$AOB_FLAGS, 2$                    ; 1560
                          F8                     07   F3 0000F 2$:      AOBLEQ  #7, FLAG, 1$                              ;
                                                 50   D4 00013          CLRL    BIT_POS                                   ; 1562
                          00        FC     A2    50   E5 00015 3$:      BBCC    BIT_POS, QA$AOB_CHECK_STATE, 4$           ; 1563
                          F7                     0E   F3 0001A 4$:      AOBLEQ  #14, BIT_POS, 3$                          ;
                                                 50   D4 0001E          CLRL    ERROR_ENTRY                               ; 1565
                                       CC A240   B4   00020 5$:         CLRW    QA$AW_OVERALL_ERROR_TABLE[ERROR_ENTRY]    ; 1566
                          F8                     0E   F3 00024          AOBLEQ  #14, ERROR_ENTRY, 5$                      ;
                                  00000000G      EF   D4 00028          CLRL    DS$GA_DS_CTRL_C_FIRST                     ; 1570
                                                 04   0002E          RET                                               ; 1571

; Routine Size:  47 bytes,    Routine Base:  CODE + 0370
```

```
;    1572  1      %SBTTL 'QA$Main'
;    1573  1
;    1574  1      GLOBAL ROUTINE QA$Main (Hook_Point) : NOVALUE =
;    1575  1
;    1576  1      !++
;    1577  1      ! FUNCTIONAL DESCRIPTION:
;    1578  1      !
;    1579  1      !       This global routine is called from different points in the Supervisor and from the DSX$Branch
;    1580  1      !       routine. It decides, based on the QA$AOB_Check_State bit vector, which check is currently being
;    1581  1      !       executed. If a check routine returns with an unsuccessful return, this will call the QA$Dump
;    1582  1      !       routine and then halt. If the diagnostic was not run with the /QA qualifier, do nothing.
;    1583  1      !
;    1584  1      ! CALLERS:
;    1585  1      !
;    1586  1      !       DSX$Branch
;    1587  1      !       CLI
;    1588  1      !       DISPAT
;    1589  1      !       ERROR
;    1590  1      !       KERNEL
;    1591  1      !       LOOP
;    1592  1      !       PARAM
;    1593  1      !       START
;    1594  1      !
;    1595  1      ! FORMAL PARAMETERS:
;    1596  1      !
;    1597  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant
;    1598  1      !       value. The constants are defined in the DSQA macro.
;    1599  1      !
;    1600  1      ! IMPLICIT INPUTS:
;    1601  1      !
;    1602  1      !       NONE
;    1603  1      !
;    1604  1      ! IMPLICIT OUTPUTS:
;    1605  1      !
;    1606  1      !       NONE
;    1607  1      !
;    1608  1      ! COMPLETION CODES:
;    1609  1      !
;    1610  1      !       NONE
;    1611  1      !
;    1612  1      ! SIDE EFFECTS:
;    1613  1      !
;    1614  1      !       NONE
;    1615  1      !
;    1616  1      !--
```

```
;    1617  2      BEGIN
;    1618  2              MACRO
;    1619  2                      !+
;    1620  2                      ! Define a macro
;    1621  2                      !
;    1622  2                      !       Check_Case (Normal_Start)
;    1623  2                      !
;    1624  2                      ! that will expand to:
;    1625  2                      !
;    1626  2                      !       [.QA$AOB_Check_State [QA$K_Normal_Start]:
;    1627  2                      !               IF (QA$Normal_Start (.Hook_Point)) THEN
;    1628  2                      !                       RETURN
;    1629  2                      !
;    1630  2                      ! for the case of the Normal Start check. That is, if the appropriate bit is set in the bit
;    1631  2                      ! vector, call the appropriate routine (corresponding to that bit) with the Hook_Point parameter
;    1632  2                      ! that was passed to QA$Main. If the routine returns successfully, return to the caller.
;    1633  2                      !-
;    1634  2                      !
;  M 1635  2                      Check_Case (Routine_Name) =
;  M 1636  2                              [ .QA$AOB_Check_State
;  M 1637  2                                      [ %NAME ('QA$K_', Routine_Name) ]
;  M 1638  2                              ]:
;  M 1639  2                                      IF (%NAME ('QA$', Routine_Name) (.Hook_Point))
;  M 1640  2                                      THEN
;  M 1641  2                                              RETURN
;    1642  2                      %;
;    1643  2                      !
;    1644  2              BUILTIN
;    1645  2                      SP;
;    1646  2                      !
;    1647  2              MAP                                                                             [02]
;    1648  2                      DS$GL_CLIBASE : BLOCK [, BYTE];                                         [02]
```

```
;   1649  2              IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN
; P 1650  2                      $DS_PRINTF ($ASCIC ('!/QA$Main Entry, Hook_Point = !UL.!/'),
;   1651  2                              .Hook_Point);
;   1652  2
;   1653  2              !+
;   1654  2              ! If not running QA, return.
;   1655  2              !-
;   1656  2
;   1657  2              IF (NOT .DSA$V_QA) THEN
;   1658  2                      RETURN;
;   1659  2
;   1660  2              !+
;   1661  2              ! Check several special cases of hook points:
;   1662  2              !
;   1663  2              !       a) If called from the START module, VRStart label, perform all the global initializations      [02]
;   1664  2              !          for this module. Do not call any check routines.                                            [02]
;   1665  2              !
;   1666  2              !       b) If called from the START module, VRReStart label, re-initialize the CLI vector and call    [02]
;   1667  2              !          the appropriate check routine. This hook-point is the starting point for each check.        [02]
;   1668  2              !
;   1669  2              !       c) If called from the START module, Before_Header area, and this is the first time that this   [02]
;   1670  2              !          hook point has been seen, then save the first and last test numbers after they have been    [02]
;   1671  2              !          set up by the START routine. Do not call any check routines.                                [02]
;   1672  2              !
;   1673  2              !       d) If called from one of the GET routines in the PARAM module, print a QA error message,    [02]
;   1674  2              !          find out what check routine is currently running, add an error to that routines total,     [02]
;   1675  2              !          and abort this diagnostic's QA. This is done here because each check routine has to make    [02]
;   1676  2              !          this check.                                                                                 [02]
;   1677  2              !
;   1678  2              !       e) If called from the midpoint of the EndPass routine in the DISPAT module, and the last    [02]
;   1679  2              !          check was executing, do things necessary to end the QA sequence of checks. That is,        [02]
;   1680  2              !          print the summary table, re-init for next time, turn the QA DSA bit off, and return to     [02]
;   1681  2              !          the EndPass routine. The act of turning the QA bit off will cause the EndPass routine      [02]
;   1682  2              !          to then branch to BEGIN, which then calls DS$Cli (effectively ending this QA sequence).    [02]
;   1683  2              !
;   1684  2              !       f) If called from the KERNEL module, Init_Context label, and if the QA debug flag is set,  [02]
;   1685  2              !          print a debugging message for me.                                                           [02]
;   1686  2              !
;   1687  2              !       g) If called either the beginning or the end of the DS_Cleanup routine in the DISPAT module,[02]
;   1688  2              !          set (or clear) the Doing_Cleanup bit in the Routine_State bitvector. This tells the check  [02]
;   1689  2              !          routines if a reported error happened during Cleanup or not.                               [02]
;   1690  2              !
```

```
;    1691  2              IF (.Hook_Point EQL DSQA$K_Start_VRStart) THEN          !                                                [02]
;    1692                  BEGIN
;    1693                  !+
;    1694                  ! Initialize QA. It should get to this hook point only when the START routine is called from     [02]
;    1695                  ! the DS$Cli routine. Set the Normal Start check bit so that the Normal Start routine is         [02]
;    1696                  ! executed first. Return to the Start module (i.e., do not execute any check routines).         [02]
;    1697                  ! NOTE: this hook point is seen only ONCE per QA sequence of checks (that is, once for each     [02]
;    1698                  ! invocation of the diagnostic with the /QA qualifier).                                          [02]
;    1699                  !-
;    1700
;    1701                  DS$GL_CLIB[ ... [CLI$L_PASS] = 1;          ! Assume one for the number of passes.                [02]
;    1702                  DS$GL_CLI[ ... [CLI$L_SUBT] = 0;           ! Start at first subtest.                             [02]
;    1703                  QA$W_Save_Test = -1;                       ! Set to -1 to indicate that it has not been saved.   [02]
;    1704                  QA$W_Save_Last = -1;                       ! Set to -1 to indicate that it has not been saved.   [02]
;    1705
;    1706                  QA$L_Saved_DSA_Flags = .DSA$GL_FLAGS;      ! Save the current DSA flag settings                  [02]
;    1707
;    1708                  DSA$V_BELL   = Off;                        ! Set BELL on error off                               [02]
;    1709                  DSA$V_HALT   = Off;                        ! Set HALT off                                        [02]
;    1710                  DSA$V_IE1    = Off;                        ! Set IE1 off                                         [02]
;    1711                  DSA$V_IE2    = Off;                        ! Set IE2 off                                         [02]
;    1712                  DSA$V_IE3    = Off;                        ! Set IE3 off                                         [02]
;    1713                  DSA$V_IES    = Off;                        ! Set IES off                                         [02]
;    1714                  DSA$V_LOOP   = Off;                        ! Set LOOP ON ERROR off                               [02]
;    1715                  DSA$V_OPER   = Off;                        ! Set OPER off                                        [02]
;    1716                  DSA$V_PROMPT = Off;                        ! Set PROMPT off                                      [02]
;    1717                  DSA$V_QUICK  = Off;                        ! Set QUICK off                                       [02]
;    1718                  DSA$V_SEARCH = Off;                        ! Set SEARCH off                                      [02]
;    1719                  DSA$V_TRACE  = On;                         ! Set TRACE flag on.                                  [02]
;    1720  3
;    1721  3               INCR State FROM 0 TO (QA$K_Numb_Routine_States - 1) DO  !                                        [02]
;    1722  3                     QA$AOB_Routine_State [.State] = Off;              !                                        [02]
;    1723  3
;    1724  3               QA$Init ();
;    1725  3               DS$GA_DS_Ctrl_C_First = QA$Control_C_Handler;          ! Enable DS catching of Control-C's.  [06]
;    1726  3               QA$AOB_Check_State [QA$K_Normal_Start] = On;
;    1727  3               RETURN                                                 !                                        [02]
;    1728  3               END
```

```
;    1729  2              ELSE IF (.Hook_Point EQL DSQA$K_Start_VRReStart) THEN          !                                                [02]
;    1730  3              BEGIN                                                         !                                                [02]
;    1731  3                      !+
;    1732  3                      ! Restore the CLI vector. This is done at the start of each check routine to protect against    [02]
;    1733  3                      ! the CLI values getting lost while a check routine is executing the diagnostic. Only restore   [02]
;    1734  3                      ! the First and Last test numbers if they have been saved (i.e., if the saved First test number [02]
;    1735  3                      ! is not -1).                                                                                    [02]
;    1736  3                      !-
;    1737  3
;    1738  3                      DS$GL_CLIBASE [CLI$L_PASS] = 1;          ! Assume one for the number of passes.                  [02]
;    1739  3                      DS$GL_CLIBASE [CLI$L_SUBT] = 0;          ! Start at first subtest.                               [02]
;    1740  3
;    1741  3                      IF (.QA$W_Save_Test NEQ -1) THEN         !                                                       [02]
;    1742  4                              BEGIN                            ! The first and last tests have been saved.             [02]
;    1743  4                              DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! Restore saved first test.             [02]
;    1744  4                              DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last    ! Restore saved last test.              [02]
;    1745  4                              END                              !                                                       [02]
;    1746  3                      END                                      !                                                       [02]
;    1747  3
;    1748  2              ELSE IF (.Hook_Point EQL DSQA$K_Start_Before_Header) THEN      !                                        [02]
;    1749  3              BEGIN
;    1750  3                      IF (.QA$W_Save_Test EQL -1) THEN         !                                                       [02]
;    1751  4                              BEGIN                            !                                                       [02]
;    1752  4                              !+
;    1753  4                              ! The first and last tests have not been saved. Save them on the first time only.       [02]
;    1754  4                              !-
;    1755  4
;    1756  4                              QA$W_Save_Test = .DS$GL_FSTTEST; ! Save first test number.                               [02]
;    1757  4                              QA$W_Save_Last = .DS$GL_LSTTEST  ! Save last test number.                                [02]
;    1758  3                              END;                             !                                                       [02]
;    1759  3                      RETURN                                   !                                                       [02]
;    1760  3              END                                              !                                                       [02]
```

```
;    1761  2              ELSE IF (.Hook_Point EQL DSQA$K_Param_DSX$GetData) OR                    !                  [02]
;    1762  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetYield) OR                  !                  [02]
;    1763  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetAddress) OR                !                  [02]
;    1764  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetLogical) OR                !                  [02]
;    1765  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetString) THEN               !                  [02]
;    1766  2                     BEGIN                                                            !                  [02]
;    1767  2                     !+                                                               !                  [02]
;    1768  2                     ! User is trying to request information from the operator. It is a QA error if no  [02]
;    1769  2                     ! default value is given.                                        !                  [02]
;    1770  2                     !-                                                               !                  [02]
;    1771  2                                                                                      !                  [02]
;    1772  2                     IF (.QA$AOB_Flags [QA$K_NoDef]) THEN                             !                  [02]
;    1773  3                             BEGIN                                                    !                  [02]
;    1774  3                             $DS_PRINTF (QA$T_Operator_Request_Message);              !                  [02]
;    1775  3                                                                                      !                  [02]
;    1776  3                             BEGIN                                                    !                  [02]
;    1777  3                             !+                                                       !                  [02]
;    1778  3                             ! Find which check routine is currently running. If none are running, it is a logic  [02]
;    1779  3                             ! error. Use the number of the first-bit-set as the QA$K_ check routine constant, and  [02]
;    1780  3                             ! add an error to this check routines summary table total.                  [02]
;    1781  3                             !-                                                       !                  [02]
;    1782  3                             LOCAL                                                    !                  [02]
;    1783  3                                     First_Bit_Set;          ! Return bit position    !                  [02]
;    1784  3                                                                                      !                  [02]
;    1785  3                             BUILTIN                                                  !                  [02]
;    1786  3                                     FFS;                    ! Find First Set bit     !                  [02]
;    1787  3                                                                                      !                  [02]
;    1788  3                             FFS (%REF (0), %REF (QA$K_Number_Of_Checks), QA$AOB_Check_State, First_Bit_Set);  !  [02]
;    1789  3                                                                                      !                  [02]
;    1790  3                             IF (.First_Bit_Set EQL QA$K_Number_Of_Checks) THEN       !                  [02]
;    1791  4                                     Logic_Error ('No check state bit set - GET routine called')  !      [02]
;    1792  3                             ELSE                                                     !                  [02]
;    1793  4                                     BEGIN                                            !                  [02]
;    1794  4                                     QA$Add_QA_Error (.First_Bit_Set);                !                  [02]
;    1795  4                                     QA$AOB_Routine_State [QA$V_Error_Found] = True   !                  [02]
;    1796  3                                     END;                                             !                  [02]
;    1797  2                             END;                                                     !                  [02]
;    1798  2                                                                                      !
;    1799  2                             QA$Abort_QA ()                  ! Abort the QA sequence  !                  [02]
;    1800  2                             END
;    1801  1                     ELSE                                                             !                  [02]
;    1802  1                     !+                                                               !                  [02]
;    1803  1                     ! There was a default value given for the GET supervisor call. Simply return.  [02]
;    1804  1                     !-                                                               !                  [02]
;    1805  1                     RETURN                                                           !                  [02]
;    1806  1              END
```

```
;    1807  2          ELSE IF (.Hook_Point EQL DSQA$K_Dispat_DSX$EndPass_Mid) THEN    ! If midpoint of EndPass routine,        [02]
;    1808  3                  BEGIN
;    1809  3                  IF (.QA$AOB_Check_State [QA$K_Last_Check]) THEN         ! If executing last check,               [02]
;    1810  4                          BEGIN                                          ! Then                                   [02]
;    1811  4                          QA$Print_Forced_Errors ();                     ! Print the forced-errors,               [02]
;    1812  4                          DS$GL_FSTTEST = .QA$W_Save_Test;               ! Restore the saved test number,         [02]
;    1813  4                          DS$GL_LSTTEST = .QA$W_Save_Last;               ! Restore the saved last test number,    [02]
;    1814  4                          QA$Print_Overall_Errors ();                    ! Print the summary table,               [02]
;    1815  4                          DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;          ! Restore the DSA flags,                 [02]
;    1816  4                          QA$Init ();                                    ! Initialize for next run,               [02]
;    1817  4                          DSA$V_QA = Off;                                ! Turn off QA bit,                       [02]
;    1818  4                          RETURN                                         ! And return to the EndPass routine.     [02]
;    1819  4                          END                                            !                                        [02]
;    1820  3                  END                                                    !                                        [02]
;    1821  3
;    1822  2          ELSE IF (.Hook_Point EQL DSQA$K_Kernel_Init_Context) THEN      !                                        [02]
;    1823  3                  BEGIN                                                  !                                        [02]
;    1824  3                  IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN                !                                        [02]
;    1825  3                          $DS_PRINTF ($ASCIC ('!/QA_Main: Init_Context done!/')); !                               [02]
;    1826  3                  RETURN
;    1827  3                  END
;    1828  3
;    1829  2          ELSE IF (.Hook_Point EQL DSQA$K_Dispat_DS_Cleanup_Bgn) THEN    !                                        [02]
;    1830  3                  BEGIN                                                  !                                        [02]
;    1831  3                  QA$AOB_Routine_State [QA$V_Doing_Cleanup] = True;      !                                        [02]
;    1832  3                  RETURN                                                 !                                        [02]
;    1833  3                  END                                                    !                                        [02]
;    183'  3
;    1835  2          ELSE IF (.Hook_Point EQL DSQA$K_Dispat_DS_Cleanup_End) THEN    !                                        [02]
;    1836  3                  BEGIN                                                  !                                        [02]
;    1837  3                  QA$AOB_Routine_State [QA$V_Doing_Cleanup] = False;     !                                        [02]
;    1838  3                  RETURN                                                 !                                        [02]
;    1839  2                  END;                                                   !                                        [02]
```

```
;    1840  2              !+
;    1841  2              ! Dispatch to the proper check routine based on what bit is set in the QA$AOB_Check_State bit vector.
;    1842  2              !-
;    1843  2
;    1844  2              SELECTONE 1 OF
;    1845  2                      SET
;    1846  2                      Check_Case (Normal_Start);
;    1847  2                      Check_Case (Multiple_Pass);
;    1848  2                      Check_Case (Loop_On_Test);
;    1849  2                      Check_Case (Loop_On_Subtest);           !                                                   [01]
;    1850  2                      Check_Case (Run_Backwards);             !                                                   [07]
;    1851  2                      [OTHERWISE]:
;    1852  3                              BEGIN
;    1853  3                              !+
;    1854  3                              ! We have found a QA internal bug. Print message for me and abort me.             [02]
;    1855  3                              !-
;    1856  3
;    1857  3                              Logic_Error ('QA$Main: No check state bit set.');       !                         [02]
;    1858  3                              DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;                   ! Restore DSA flags
;    1859  3                              QA$Init ();                                             ! Initialize for next run
;    1860  3                              DSA$V_QA = Off;                                         ! Turn off QA
;    1861  3                              DSX$Abort ();                                           ! Good way to stop QA.
;    1862  3                              RETURN
;    1863  2                              END;
;    1864  2                      TES;
;    1865  2
;    1866  2              !+
;    1867  2              ! If it gets to here, then a check routine failed. This implies that we detected a QA error.      [02]
;    1868  2              ! Abort me and the diagnostic.                                                                     [02]
;    1869  2              !-
;    1870  2
;    1871  2              QA$Abort_QA ()                          !                                                         [02]
;    1872  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

72 74 6E 45 20 6E 69 61 4D 24 41 51 2F 21 24  0021F P.AAN:   .ASCII  \$!/QA$Main Entry, Hook_Point = !UL.!/\
3D 20 74 6E 69 6F 50 5F 6B 6F 6F 48 20 2C 79  0022E
                        2F 21 2E 4C 55 21 20  0023D
61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 44  00244 P.AAO:   .ASCII  \D!/**QA Internal Error: No check state b\
65 68 63 20 6F 4E 20 3A 72 6F 72 72 45 20 6C  00253
                  62 20 65 74 61 74 73 20 6B  00262
6F 72 20 54 45 47 20 2D 20 74 65 73 20 74 69  0026C         .ASCII  \it set - GET routine called!/\
   2F 21 64 65 6C 6C 61 63 20 65 6E 69 74 75  0027B
69 6E 49 20 3A 6E 69 61 4D 5F 41 51 2F 21 1E  00289 P.AAP:   .ASCII  <30>\!/QA_Main: Init_Context done!/\
21 65 6E 6F 64 20 74 78 65 74 6E 6F 43 5F 74  00298
                                       2F 2A  002A7
61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 39  002A6 P.AAQ:   .ASCII  \9!/**QA Internal Error: QA$Main: No chec\
69 61 4D 24 41 51 20 3A 72 6F 72 72 45 20 6C  002B7
                  63 65 68 63 20 6F 4E 20 3A  002C6
74 65 73 20 74 69 62 20 65 74 61 74 73 20 6B  002D0         .ASCII  \k state bit set.!/\
                                    2F 21 2E  002DF

                                                ES=                     P.AAO
                                                ES=                     P.AAQ
```

```
                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                03FC 0000               .ENTRY  QA$MAIN, Save R2,R3,R4,R5,R6,R7,R8,R9       ; 1574
                                    59 00C00000G EF   9E 00002          MOVAB   DS$GL_LSTTEST, R9                           ;
                                    58 00000000G EF   9E 00009          MOVAB   DS$GL_FSTTEST, R8                           ;
                                    57 00000000' EF   9E 00010          MOVAB   P.AAN, R7                                   ;
                                    56 00000000G 9F   9E 00017          MOVAB   @#DS$PRINTF, R6                             ;
                                    55 00000000G EF   9E 0001E          MOVAB   DS$GL_CLIBASE+44, R5                        ;
                                    54       A9  AF   9E 00025          MOVAB   QA$INIT, R4                                 ;
                                    53 00000000' EF   9E 00029          MOVAB   QA$AOB_CHECK_STATE, R3                      ;
                          08    04  A3                07  E1 00030          BBC     #1, QA$AOB_FLAGS, 1$                ; 1649
                                        04  AC  DD 00035          PUSHL   HOOK_POINT                                  ; 1651
                                        57  DD 00038          PUSHL   R7                                          ;
                                        02  FB 0003A          CALLS   #2, DS$PRINTF                               ;
                                    66                  0000FE01  9F   95 0003D 1$:     TSTB    @#^X0000FE01                ; 1657
                                        01  19 00043          BLSS    2$                                          ;
                                        04  00045          RET                                                 ;
                                    52       04  AC  D0 00046 2$:     MOVL    HOOK_POINT, R2                      ; 1691
                                    16           22  D1 0004A          CMPL    R2, #22                             ;
                                        3D  12 0004D          BNEQ    5$                                          ;
                                    65           01  D0 0004F          MOVL    #1, DS$GL_CLIBASE+44                ; 1701
                                            FC  A5  D4 00052          CLRL    DS$GL_CLIBASE+40                    ; 1702
                                  F2    A3           01  CE 00055          MNEGL   #1, QA$W_SAVE_TEST                  ; 1703
                                  F8    A3 0000FE00  9F  D0 00059          MOVL    @#^X0000FE00, QA$L_SAVED_DSA_FLAGS  ; 1706
                          0000FE00      9F      71FD  8F  AA 00061          BICW2   #29181, @#^X0000FE00                ; 1718
                          0000FE01      9F            04  88 0006A          BISB2   #4, @#^X0000FE01                    ; 1719
                                        50  D4 00071          CLRL    STATE                                       ; 1721
                      00        FC      A7            50  E5 00073 3$:     BBCC    STATE, QA$AOB_ROUTINE_STATE, 4$     ; 1722
                      F7                50            03  F3 00078 4$:     AOBLEQ  #3, STATE, 3$                       ;
                                        64  00  FB 0007C          CALLS   #0, QA$INIT                                 ; 1724
                          00000000G     EF      FE1F  C4  9E 0007F          MOVAB   QA$CONTROL_C_HANDLER, DS$GA_DS_CTRL_C_FIRST  ; 1725
                                        63           01  88 00088          BISB2   #1, QA$AOB_CHECK_STATE              ; 1726
                                        04  00088          RET                                                 ; 1692
                                        12           52  D1 0008C 5$:     CMPL    R2, #18                             ; 1729
                                        1A  12 0008F          BNEQ    6$                                          ;
                                        65           01  D0 00091          MOVL    #1, DS$GL_CLIBASE+44                ; 1738
                                                FC  A5  D4 00094          CLRL    DS$GL_CLIBASE+40                    ; 1739
                                    FFFF    8F      F2  A3  B1 00097          CMPW    QA$W_SAVE_TEST, #-1                 ; 1741
                                        6C  13 0009D          BEQL    13$                                         ;
                                      F4    A5      F2  A3  32 000A1          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+32    ; 1743
                                      F8    A5      F4  A3  32 000A4          CVTWL   QA$W_SAVE_LAST, DS$GL_CLIBASE+36    ; 1744
                                        60  11 000A9          BRB     13$                                         ; 1741
                                        17           52  D1 000AB 6$:     CMPL    R2, #23                             ; 1748
                                        12  12 000AE          BNEQ    8$                                          ;
                                    FFFF    8F      F2  A3  B1 000B0          CMPW    QA$W_SAVE_TEST, #-1                 ; 1750
                                        01  13 000B6          BEQL    7$                                          ;
                                        04  000B8          RET                                                 ;
                                      F2    A3          68  B0 000B9 7$:     MOVW    DS$GL_FSTTEST, QA$W_SAVE_TEST       ; 1756
                                      F4    A3          69  B0 000BD          MOVW    DS$GL_LSTTEST, QA$W_SAVE_LAST       ; 1757
                                        04  000C1          RET                                                 ; 1749
                                        0C           52  D1 000C2 8$:     CMPL    R2, #12                             ; 1761
                                        14  13 000C5          BEQL    9$                                          ;
                                        0D           52  D1 000C7          CMPL    R2, #13                             ; 1762
                                        0F  13 000CA          BEQL    9$                                          ;
                                        0E           52  D1 000CC          CMPL    R2, #14                             ; 1763
```

```
                                      0A  13 000CF        BEQL    9$                                          ;
                          0F          52  D1 000D1        CMPL    R2, #15                                     ; 1764
                                      05  13 000D4        BEQL    9$                                          ;
                          10          52  D1 000D6        CMPL    R2, #16                                     ; 1765
                                      32  12 000D9        BNEQ    14$                                         ;
                          01      04  A3  E8 000DB 9$:    BLBS    QA$AOB_FLAGS, 10$                           ; 1772
                                          04 000DF        RET                                                 ;
                          00000000G   EF  9F 000E0 10$:   PUSHAB  QA$T_OPERATOR_REQUEST_MESSAGE               ; 1774
                          66          01  FB 000E6        CALLS   #1, DS$PRINTF                               ;
       50          63     0F          00  EA 000E9        FFS     #0, #15, QA$AOB_CHECK_STATE, FIRST_BIT_SET  ; 1788
                          0F          50  D1 000EE        CMPL    FIRST_BIT_SET, #15                          ; 1790
                                      08  12 000F1        BNEQ    11$                                         ;
                                  25  A7  9F 000F3        PUSHAB  ES                                          ; 1791
                          66          01  FB 000F6        CALLS   #1, DS$PRINTF                               ;
                                      0B  11 000F9        BRB     12$                                         ; 1790
                                      50  DD 000FB 11$:   PUSHL   FIRST_BIT_SET                               ; 1794
                   FE11   C4          01  FB 000FD        CALLS   #1, QA$ADD_QA_ERROR                         ;
                     FC   A3          02  88 00102        BISB2   #2, QA$AOB_ROUTINE_STATE                    ; 1795
                   FDDC   C4          00  FB 00106 12$:   CALLS   #0, QA$ABORT_QA                             ; 1799
                                      58  11 0010B 13$:   BRB     19$                                         ;
                                      52  D5 0010D 14$:   TSTL    R2                                          ; 1807
                                      2E  12 0010F        BNEQ    15$                                         ;
                   50          63     04  E1 00111        BBC     #4, QA$AOB_CHECK_STATE, 19$                 ; 1809
               00000000V   EF         00  FB 00115        CALLS   #0, QA$PRINT_FORCED_ERRORS                  ; 1811
                           68     F2  A3  32 0011C        CVTWL   QA$W_SAVE_TEST, DS$GL_FSTTEST               ; 1812
                           69     F4  A3  32 00120        CVTWL   QA$W_SAVE_LAST, DS$GL_LSTTEST               ; 1813
               00000000V   EF         00  FB 00124        CALLS   #0, QA$PRINT_OVERALL_ERRORS                 ; 1814
               0000FE00    9F     F8  A3  D0 0012B        MOVL    QA$L_SAVED_DSA_FLAGS, @#^X0000FE00          ; 1815
                           64         00  FB 00133        CALLS   #0, QA$INIT                                 ; 1816
               0000FE01    9F     80  8F  8A 00136        BICB2   #128, @#^X0000FE01                          ; 1817
                                          04 0013E        RET                                                 ; 1810
                          08          52  D1 0013F 15$:   CMPL    R2, #8                                      ; 1822
                                      0D  12 00142        BNEQ    17$                                         ;
                   01          04     A3  01  E0 00144    BBS     #1, QA$AOB_FLAGS, 16$                       ; 1824
                                          04 00149        RET                                                 ;
                                  6A  A7  9F 0014A 16$:   PUSHAB  P.AAP                                       ; 1825
                          66          01  FB 0014D        CALLS   #1, DS$PRINTF                               ;
                                          04 00150        RET                                                 ; 1823
                          03          52  D1 00151 17$:   CMPL    R2, #3                                      ; 1829
                                      05  12 00154        BNEQ    18$                                         ;
                     FC   A3          04  88 00156        BISB2   #4, QA$AOB_ROUTINE_STATE                    ; 1831
                                          04 0015A        RET                                                 ; 1830
                          04          52  D1 0015B 18$:   CMPL    R2, #4                                      ; 1835
                                      05  12 0015E        BNEQ    19$                                         ;
                     FC   A3          04  8A 00160        BICB2   #4, QA$AOB_ROUTINE_STATE                    ; 1837
                                          04 00164        RET                                                 ; 1836
                          0B          63  E9 00165 19$:   BLBC    QA$AOB_CHECK_STATE, 20$                     ; 1846
                                      52  DD 00168        PUSHL   R2                                          ;
               00000000G   EF         01  FB 0016A        CALLS   #1, QA$NORMAL_START                         ;
                                      3A  11 00171        BRB     24$                                         ;
                   0B          63     01  E1 00173 20$:   BBC     #1, QA$AOB_CHECK_STATE, 21$                 ; 1847
                                      52  DD 00177        PUSHL   R2                                          ;
               00000000G   EF         01  FB 00179        CALLS   #1, QA$MULTIPLE_PASS                        ;
                                      2B  11 00180        BRB     24$                                         ;
                   0B          63     02  E1 00182 21$:   BBC     #2, QA$AOB_CHECK_STATE, 22$                 ; 1848
                                      52  DD 00186        PUSHL   R2                                          ;
               00000000G   EF         01  FB 00188        CALLS   #1, QA$LOOP_ON_TEST                         ;
```

```
                                        1C  11 0018F          BRB     24$                                      ;
          0B          63                03  E1 00191 22$:     BBC     #3, QA$AOB_CHECK_STATE, 23$              ; 1849
                                        52  DD 00195          PUSHL   R2                                       ;
            00000000G EF                01  FB 00197          CALLS   #1, QA$LOOP_ON_SUBTEST                   ;
                                        0D  11 0019E          BRB     24$                                      ;
          0D          63                04  E1 001A0 23$:     BBC     #4, QA$AOB_CHECK_STATE, 25$              ; 1850
                                        52  DD 001A4          PUSHL   R2                                       ;
            00000000G EF                01  FB 001A6          CALLS   #1, QA$RUN_BACKWARDS                     ;
                      23                50  E9 001AD 24$:     BLBC    R0, 26$                                  ;
                                            04 001B0          RET                                              ;
                               0089     C7  9F 001B1 25$:     PUSHAB  ES                                       ; 1857
                      66                01  FB 001B5          CALLS   #1, DS$PRINTF                            ;
            0000FE00  9F         F8     A3  D0 001B8          MOVL    QA$L_SAVED_DSA_FLAGS, @#^X0000FE00       ; 1858
                      64                00  FB 001C0          CALLS   #0, QA$INIT                              ; 1859
            0000FE01  9F         80     8F  8A 001C3          BICB2   #128, @#^X0000FE01                       ; 1860
            00000000G EF                00  FB 001CB          CALLS   #0, DSX$ABORT                            ; 1861
                                            04 001D2          RET                                              ; 1852
                 FDDC C4                00  FB 001D3 26$:     CALLS   #0, QA$ABORT_QA                          ; 1871
                                            04 001D8          RET                                              ; 1872

; Routine Size:  473 bytes,    Routine Base:  CODE + 039F
```

```
;    1873  1     %SBTTL 'QA$Print_Forced_Errors'
;    1874  1
;    1875  1     GLOBAL ROUTINE QA$Print_Forced_Errors : NOVALUE =       !                                        ALL [02]
;    1876  1
;    1877  1     !++
;    1878  1     ! FUNCTIONAL DESCRIPTION:
;    1879  1     !
;    1880  1     !       This routine prints the Forced-Errors Summary Table, which gives the number of forced errors for the
;    1881  1     !       three Error checks.
;    1882  1     !
;    1883  1     ! CALLER(S):
;    1884  1     !
;    1885  1     !       The Error check routines.
;    1886  1     !
;    1887  1     ! FORMAL PARAMETERS:
;    1888  1     !
;    1889  1     !       NONE
;    1890  1     !
;    1891  1     ! IMPLICIT INPUTS:
;    1892  1     !
;    1893  1     !
;    1894  1     !
;    1895  1     ! IMPLICIT OUTPUTS:
;    1896  1     !
;    1897  1     !       Forced-Errors Summary Table printout.
;    1898  1     !
;    1899  1     ! COMPLETION CODES:
;    1900  1     !
;    1901  1     !       NONE
;    1902  1     !
;    1903  1     ! SIDE EFFECTS:
;    1904  1     !
;    1905  1     !       NONE
;    1906  1     !
;    1907  1     !--
```

```
;    1908  1     %SBTTL 'Dispose'
;    1909  2     BEGIN                                                                   ! Print_Forced_Errors Routine
;    1910  2     !
;    1911  2     !++
;    1912  2     ! FUNCTIONAL DESCRIPTION:
;    1913  2     !
;    1914  2     !       This routine disposes of a 16-byte vector.
;    1915  2     !
;    1916  2     ! CALLER (S):
;    1917  2     !
;    1918  2     !       QA$Print_Forced_Errors
;    1919  2     !
;    1920  2     ! FORMAL PARAMETERS:
;    1921  2     !
;    1922  2     !       Record_Ptr - Contains the address of the 16-byte vector.
;    1923  2     !
;    1924  2     ! IMPLICIT INPUTS:
;    1925  2     !
;    1926  2     !       NONE
;    1927  2     !
;    1928  2     ! IMPLICIT OUTPUTS:
;    1929  2     !
;    1930  2     !       NONE
;    1931  2     !
;    1932  2     ! COMPLETION CODES:
;    1933  2     !
;    1934  2     !       NONE
;    1935  2     !
;    1936  2     ! SIDE EFFECTS:
;    1937  2     !
;    1938  2     !       NONE
;    1939  2     !
;    1940  2     !--
```

```
;    1941  2      !
;    1942  2      !
;    1943  2      ! ROUTINE Dispose (Record_Ptr) : NOVALUE =
;    1944  2      !       BEGIN
;    1945  2      !               MAP
;    1946  2      !                       Record_Ptr : REF VECTOR [, WORD];               ! Treat the Record_Ptr as a vector of words
;    1947  2      !
;    1948  2      !               Record_Ptr [4] = QA$K_Forced_Error_Record_Size;         ! Put in the size of the data structure
;    1949  2      !               EXE$DeaNonPaged (.Record_Ptr);                          ! Deallocate it
;    1950  2      !       END;
```

```
;    1951  2
;    1952  2        %SBTTL 'QA$Print_Forced_Errors'
;    1953  2                !+
;    1954  2                ! The are the strings that get printed in the table.
;    1955  2                !-
;    1956  2
;    1957  2                BIND
;    1958  2                        Header_Line_1 =                                                         [04]
;    1959  2                                $ASCIC ('!^!/!/!#* Summary of Forced-Errors!/'),                [04]
;    1960  2
;    1961  2                        Header_Line_2 =                                                         [04]
;    1962  2                                $ASCIC ('!#* ------------------------!/!/'),                    [04]
;    1963  2
;    1964  2                        Test_Subtest_Line_1 =
;    1965  2                                $ASCIC ('Test !3UW   Subtest !3UL!/'),
;    1966  2
;    1967  2                        Test_Subtest_Line_2 =
;    1968  2                                $ASCIC ('!8*-   !11*-!/'),
;    1969  2
;    1970  2                        Error_Count_Line =
;    1971  2                                $ASCIC ('!5UW-!4<!UL!> '),
;    1972  2
;    1973  2                        Eol_Line =
;    1974  2                                $ASCIC ('!/');
;    1975  2
;    1976  2                LITERAL
;    1977  2                        QA$K_Error_Count_Line_Len = 11,         ! The length (in chars) of the          [04]
;    1978  2                                                                ! Error_Count_Line line.                [04]
;    1979  2
;    1980  2                        QA$K_Minimum_Width = 24;                ! The minimum width that can be used to print
;    1981  2                                                                ! the forced-error table. Currently this is the
;    1982  2                                                                ! length of the Header_Line.
;    1983  2
;    1984  2                LOCAL
;    1985  2                        Number_Of_Error_Counts : WORD,          ! The number of error number/error counts to
;    1986  2                                                                ! print per line.
;    1987  2
;    1988  2                        Number_Printed : WORD,                  ! The running total of the number of error
;    1989  2                                                                ! number/error counts printed per line.
;    1990  2
;    1991  2                        Record_Ptr,                             ! The pointer that is used to acess all the
;    1992  2                                                                ! records in the linked list.
;    1993  2
;    1994  2                        Temp_Ptr,                               ! The pointer to dispose.
;    1995  2
;    1996  2                        Width : WORD,                           ! The maximum width of the output line.  [04]
;    1997  2
;    1998  2                        Test : WORD,                            ! The Test_Number from the forced-error record.
;    1999  2
;    2000  2                        Subtest;                                ! The Subtest_Number from the forced-error record.
```

```
;    2001  2        !
;    2002  2        !       Width                    = MAXU (QA$K_Minimum_Width, .DS$GB_WIDTH);      !                    [0,]
;    2003  2        !       Number_Of_Error_Counts = .Width  / QA$K_Error_Count_Line_Len;            !                    [04]
;    2004  2        !
;    2005  2        !       $DS_PRINTF (Header_Line_1, ((.Width - QA$K_Minimum_Width + 1) / 2));     !                    [04]
;    2006  2        !       $DS_PRINTF (Header_Line_2, ((.Width - QA$K_Minimum_Width + 1) / 2));     !                    [04]
;    2007  2        !
;    2008  2        !       !+
;    2009  2        !       ! If it exists, skip the first null record and dispose of it.
;    2010  2        !       !-
;    2011  2        !
;    2012  2        !       Record_Ptr = .QA$A_Error_List_Head;
;    2013  2        !
;    2014  2        !       IF (.Record_Ptr NEQ QA$K_Nil) THEN
;    2015  2        !               BEGIN
;    2016  2        !                       MAP
;    2017  2        !                               QA$A_Error_List_Head : Error_Record;
;    2018  2        !
;    2019  2        !                       QA$A_Error_List_Head = .QA$A_Error_List_Head [QA$VA_Next_Error];
;    2020  2        !                       Dispose (.Record_Ptr);
;    2021  2        !                       Record_Ptr = .QA$A_Error_List_Head
;    2022  2        !               END;
```

```
;    2023  2     !
;    2024  2     !       WHILE (.QA$A_Error_List_Head NEQ QA$K_Nil) DO
;    2025  2     !               BEGIN
;    2026  2     !               MAP
;    2027  2     !                       Record_Ptr : Error_Record;
;    2028  2     !
;    2029  2     !               Test           = .Record_Ptr [QA$VW_Test_Number];
;    2030  2     !               Subtest        = .Record_Ptr [QA$VL_Subtest_Number];
;    2031  2     !               Number_Printed = 0;
;    2032  2     !
;    2033  2     !               $DS_PRINTF (Test_Subtest_Line_1, .Test, .Subtest);
;    2034  2     !               $DS_PRINTF (Test_Subtest_Line_2);
;    2035  2     !
;    2036  2     !               DO
;    2037  2     !                       BEGIN
;    2038  2     !                       $DS_PRINTF (Error_Count_Line,
;    2039  2     !                                   .Record_Ptr [QA$VW_Error_Number],
;    2040  2     !                                   .Record_Ptr [QA$VL_Error_Count]);
;    2041  2     !
;    2042  2     !                       IF ((Number_Printed = .Number_Printed + 1) EQL .Number_Of_Error_Counts) THEN
;    2043  2     !                               BEGIN
;    2044  2     !                               Number_Printed = 0;
;    2045  2     !                               $DS_PRINTF (Eol_Line);
;    2046  2     !                               END;
;    2047  2     !
;    2048  2     !                       Temp_Ptr             = .Record_Ptr;
;    2049  2     !                       Record_Ptr           = .Record_Ptr [QA$VA_Next_Error];
;    2050  2     !                       QA$A_Error_List_Head = .Record_Ptr;
;    2051  2     !
;    2052  2     !                       Dispose (.Temp_Ptr);
;    2053  2     !
;    2054  2     !                       IF (.Record_Ptr EQL QA$K_Nil) THEN
;    2055  2     !                               EXITLOOP;
;    2056  2     !                       END
;    2057  2     !               UNTIL ((.Test NEQ .Record_Ptr [QA$VW_Test_Number]) OR
;    2058  2     !                      (.Subtest NEQ .Record_Ptr [QA$VL_Subtest_Number]));
;    2059  2     !
;    2060  2     !               $DS_PRINTF (Eol_Line);
;    2061  2     !               IF (.Number_Printed NEQ 0) THEN
;    2062  2     !                       $DS_PRINTF (Eol_Line)
;    2063  2     !               END;
;    2064  2     !
;    2065  2     !       QA$A_Prev_Test_Tail  = QA$K_Nil;
;    2066  2     !       QA$A_Error_List_Head = QA$K_Nil;
;    2067  2     !
;    2068  2     !               1;                                                                              !       [07]
;    2069  2
;    2070  1     !       END;


                                        0000 00000          .ENTRY  QA$PRINT_FORCED_ERRORS, Save nothin         ; 1875
                                          04 00002          RET                                                 ; 2070

; Routine Size:  3 bytes,    Routine Base:  CODE + 0578
```

```
;    2071  1     %SBTTL 'QA$Print_Overall_Errors'
;    2072  1
;    2073  1     ROUTINE QA$Print_Overall_Errors : NOVALUE =
;    2074  1
;    2075  1     !++
;    2076  1     ! FUNCTIONAL DESCRIPTION:
;    2077  1     !
;    2078  1     !       This routine prints the Overall Error Summary Table, which summarizes the results for one QA execution
;    2079  1     !       of the diagnostic.
;    2080  1     !
;    2081  1     ! CALLER(S):
;    2082  1     !
;    2083  1     !       QA$Main
;    2084  1     !
;    2085  1     ! FORMAL PARAMETERS:
;    2086  1     !
;    2087  1     !       NONE
;    2088  1     !
;    2089  1     ! IMPLICIT INPUTS:
;    2090  1     !
;    2091  1     !       QA$W_MultiplePass         - The number of passes to make for the Multiple Pass Check.
;    2092  1     !       QA$W_TestLoops            - The number of loops for the Infinite Loop-On-Test Check.
;    2093  1     !       QA$W_SubtestLoops         - The number of loops for the Infinite Loop-On-Subtest Check.
;    2094  1     !       QA$AW_Overall_Error_Table - The number of errors that occurred in each check.
;    2095  1     !
;    2096  1     ! IMPLICIT OUTPUTS:
;    2097  1     !
;    2098  1     !       Overall Error Summary Table printout.
;    2099  1     !
;    2100  1     ! COMPLETION CODES:
;    2101  1     !
;    2102  1     !       NONE
;    2103  1     !
;    2104  1     ! SIDE EFFECTS:
;    2105  1     !
;    2106  1     !       NONE
;    2107  1     !
;    2108  1     !--
```

```
;    2109  2      BEGIN
;    2110  2
;    2111  2              !+
;    2112  2              ! The are the strings that get printed in each line of the Overall Error Summary Table.
;    2113  2              !-
;    2114  2
;    2115  2              BIND
;    2116  2                      Header_Line_1 =                                                 !                       [04]
;    2117  2                              $ASCIC ('!^!/!/!11* Overall Error Summary Table!/'),    !                       [04]
;    2118  2
;    2119  2                      Header_Line_2 =                                                 !                       [04]
;    2120  2                              $ASCIC ('!11* ----------------------------!/'),         !                       [04]
;    2121  2
;    2122  2                      Header_Line_3 =                                                 !                       [04]
;    2123  2                              $ASCIC ('QA Checklist Item!16* Number of Errors!/'),
;    2124  2
;    2125  2                      Header_Line_4 =                                                 !                       [04]
;    2126  2                              $ASCIC ('!17*-!16* !16*-!/'),
;    2127  2
;    2128  2                      NS_Line =
;    2129  2                              $ASCIC ('!41<Normal Start!50*.!>!UW!/'),
;    2130  2
;    2131  2                      MP_Line =
;    2132  2                              $ASCIC ('!41<Multiple Pass (!UW passes)!50*.!>!UW!/'),
;    2133  2
;    2134  2                      LT_Line =
;    2135  2                              $ASCIC ('!41<Infinite Loop-On-Test (!UW passes)!50*.!>!UW!/'),
;    2136  2
;    2137  2                      LS_Line =                                                       !                       [01]
;    2138  2                              $ASCIC ('!41<Infinite Loop-On-Subtest (!UW passes)!50*.!>!UW!/'),       !       [01]
;    2139  2
;    2140  2                      RB_Line =
;    2141  2                              $ASCIC ('!41<Run Tests Backwards!50*.!>!UW!/'),
;    2142  2
;    2143  2                      OK_Line =                                                       !                       [01]
;    2144  2                              $ASCIC ('!/** QA ** !16AC passed Auto-QA portion of Quality Assurance Checklist!/');
```

```
;    2145  2              !+                                                                                                    [02]
;    2146  2              ! If the /TEST qualifier was given, do not print this table out. This prevents someone from running QA  [02]
;    2147  2              ! on only a small portion of the diagnostic, and having that portion work. The following must therefore [02]
;    2148  2              ! be true:                                                                                            [02]
;    2149  2              !       .DS$GL_FSTTEST = 1                                                                            [02]
;    2150  2              !       .DS$GL_LSTTEST = ..L$A_TSTCNT                                                                 [02]
;    2151  2              ! If either one of these is false, simply return. Note: accept the following if a diagnsotic contains   [02]
;    2152  2              ! x tests: /TEST:1:x.                                                                                 [02]
;    2153  2              !-                                                                                                    [02]
;    2154  2                                                                                                                    [02]
;    2155  2              IF ((.DS$GL_FSTTEST NEQ 1) OR (.DS$GL_LSTTEST NEQ ..L$A_TSTCNT)) THEN   !                             [02]
;    2156  2                      RETURN;                                                         !                             [02]
;    2157  2                                                                                                                    [02]
;    2158  2              !+                                                                                                    [02]
;    2159  2              ! Print header lines.                                                                                 [02]
;    2160  2              !-                                                                                                    [02]
;    2161  2                                                                                                                    [02]
;    2162  2              $DS_PRINTF (Header_Line_1);                                             !                             [04]
;    2163  2              $DS_PRINTF (Header_Line_2);                                             !                             [04]
;    2164  2              $DS_PRINTF (Header_Line_3);                                             !                             [04]
;    2165  2              $DS_PRINTF (Header_Line_4);                                             !                             [04]
;    2166  2                                                                                                                    [04]
;    2167  2              !+                                                                                                    [04]
;    2168  2              ! Print each line for the checks, until a non-zero error count is                                     [04]
;    2169  2              ! reached, or until the last check. If all the checks were non-zero,                                  [04]
;    2170  2              ! print the "passed QA" message.                                                                      [04]
;    2171  2              !-                                                                                                    [04]
```

```
;    2172  2              !+
;    2173  2              ! Print and check Normal Start errors
;    2174  2              !-
;    2175  2
;  P 2176  2              $DS_PRINTF (NS_Line,
;    2177  2                      .QA$AW_Overall_Error_Table [QA$K_Normal_Start]);
;    2178  2
;    2179  2              IF (.QA$AW_Overall_Error_Table [QA$K_Normal_Start] NEQ 0) THEN
;    2180  2                      RETURN;
;    2181  2
;    2182  2              !+
;    2183  2              ! Print and check Multiple Pass errors
;    2184  2              !-
;    2185  2
;  P 2186  2              $DS_PRINTF (MP_Line, .QA$W_MultiplePass,
;    2187  2                      .QA$AW_Overall_Error_Table [QA$K_Multiple_Pass]);
;    2188  2
;    2189  2              IF (.QA$AW_Overall_Error_Table [QA$K_Multiple_Pass] NEQ 0) THEN
;    2190  2                      RETURN;
;    2191  2
;    2192  2              !+
;    2193  2              ! Print and check Loop-On-Test erros
;    2194  2              !-
;    2195  2
;  P 2196  2              $DS_PRINTF (LT_Line, .QA$W_TestLoops,
;    2197  2                      .QA$AW_Overall_Error_Table [QA$K_Loop_On_Test]);
;    2198  2
;    2199  2              IF (.QA$AW_Overall_Error_Table [QA$K_Loop_On_Test] NEQ 0) THEN
;    2200  2                      RETURN;
;    2201  2
;    2202  2              !+                                                                      [01]
;    2203  2              ! Print and check Loop-On-Subtest errors                                 [01]
;    2204  2              !-                                                                      [01]
;    2205  2
;  o 2206  2              $DS_PRINTF (LS_Line, .QA$W_SubtestLoops,                                [01]
;    2207  2                      .QA$AW_Overall_Error_Table [QA$K_Loop_On_Subtest]);              [01]
;    2208  2
;    2209  2              IF (.QA$AW_Overall_Error_Table [QA$K_Loop_On_Subtest] NEQ 0) THEN       [01]
;    2210  2                      RETURN;                                                         [01]
```

```
;    2211  2              !+
;    2212  2              ! Print and check Run Tests Backwards errors
;    2213  2              !-
;    2214  2
; P  2215  2              $DS_PRINTF (RB_Line,
;    2216  2                      .QA$AW_Overall_Error_Table [QA$K_Run_Backwards]);
;    2217  2
;    2218  2              IF (.QA$AW_Overall_Error_Table [QA$K_Run_Backwards] NEQ 0) THEN
;    2219  2                      RETURN;
;    2220  2
;    2221  2              $DS_PRINTF (OK_Line, .L$A_NAME);
;    2222  2
;    2223  2              RETURN
;    2224  1          END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

65 76 4F 20 2A 31 31 21 2F 21 2F 21 5E 21 28 002E2 P.AAR:  .ASCII  \(!^!/!/!11* Overall Error Summary Table!\  ;
6D 6D 75 53 20 72 6F 72 72 45 20 6C 6C 61 72 002F1                                                          ;
                 21 65 6C 62 61 54 20 79 72 61 00300                                                        ;
                                       2F 0030A          .ASCII  \/\                                       ;
2D 2D 2D 2D 2D 2D 2D 2D 2D 20 2A 31 31 21 22 0030B P.AAS:  .ASCII  \"!11* ----------------------------!/\    ;
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0031A                                                          ;
                                    2F 21 2D 2D 2D 00329                                                    ;
49 20 74 73 69 6C 6B 63 65 68 43 20 41 51 28 0032E P.AAT:  .ASCII  \(QA Checklist Item!16* Number of Errors!\  ;
20 72 65 62 6D 75 4E 20 2A 36 31 21 6D 65 74 0033D                                                          ;
                 21 73 72 6F 72 72 45 20 66 6F 00034C                                                       ;
                                       2F 00356          .ASCII  \/\                                       ;
2A 36 31 21 20 2A 36 31 21 2D 2A 37 31 21 11 00357 P.AAU:  .ASCII  <17>\!17*-!16* !16*-!/\                  ;
                                    2F 21 2D 00366                                                          ;
61 74 53 20 6C 61 6D 72 6F 4E 3C 31 34 21 1C 00369 P.AAV:  .ASCII  <28>\!41<Normal Start!50*.!>!UW!/\       ;
   2F 21 57 55 21 3E 21 2E 2A 30 35 21 74 72 00378                                                          ;
50 20 65 6C 70 69 74 6C 75 4D 3C 31 34 21 2A 00386 P.AAW:  .ASCII  \*!41<Multiple Pass (!UW passes)!50*.!>!U\  ;
73 65 73 73 61 70 20 57 55 21 28 20 73 73 61 00395                                                          ;
                 55 21 3E 21 2E 2A 30 35 21 29 003A4                                                        ;
                                    2F 21 57 003AE          .ASCII  \W!/\                                   ;
4C 20 65 74 69 6E 69 66 6E 49 3C 31 34 21 32 003B1 P.AAX:  .ASCII  \2!41<Infinite Loop-On-Test (!UW passes)!\ ;
55 21 28 20 74 73 65 54 2D 6E 4F 2D 70 6F 6F 003C0                                                          ;
                 21 29 73 65 73 73 61 70 20 57 003CF                                                        ;
                    2F 21 57 55 21 3E 21 2E 2A 30 35 003D9          .ASCII  \50* !>!UW!/\                   ;
4C 20 65 74 69 6E 69 66 6E 49 3C 31 34 21 35 003E4 P.AAY:  .ASCII  \5!41<Infinite Loop-On-Subtest (!UW passe\ ;
20 74 73 65 74 62 75 53 2D 6E 4F 2D 70 6F 6F 003F3                                                          ;
                 65 73 73 61 70 20 57 55 21 28 00402                                                        ;
   2F 21 57 55 21 3E 21 2E 2A 30 35 21 29 73 00040C          .ASCII  \s)!50*.!>!UW!/\                        ;
20 73 74 73 65 54 20 6E 75 52 3C 31 34 21 23 0041A P.AAZ:  .ASCII  \#!41<Run Tests Backwards!50*.!>!UW!/\    ;
21 2E 2A 30 35 21 73 64 72 61 77 6B 63 61 42 00429                                                          ;
                                    2F 21 57 55 21 3E 00438                                                 ;
36 31 21 20 2A 2A 20 41 51 20 2A 2A 2F 21 48 00043E P.ABA:  .ASCII  \H!/** QA ** !16AC passed Auto-QA portion\ ;
2D 6F 74 75 41 20 64 65 73 73 61 70 20 43 41 00044D                                                         ;
                 6E 6F 69 74 72 6F 70 20 41 51 00045C                                                       ;
73 73 41 20 79 74 69 6C 61 75 51 20 66 6F 20 00466          .ASCII  \ of Quality Assurance Checklist!/\      ;
73 69 6C 6B 63 65 68 43 20 65 63 6E 61 72 75 00475                                                          ;
                                    2F 21 74 00484                                                          ;

                                                    HEADER_LINE_1=      P.AAR
```

```
                                HEADER_LINE_2=      P.AAS
                                HEADER_LINE_3=      P.AAT
                                HEADER_LINE_4=      P.AAU
                                NS_LINE=            P.AAV
                                MP_LINE=            P.AAW
                                LT_LINE=            P.AAX
                                LS_LINE=            P.AAY
                                RB_LINE=            P.AAZ
                                OK_LINE=            P.ABA


                                        .PSECT  CODE,NOWRT,  SHR,2

                                001C 00000 QA$PRINT_OVERALL_ERRORS:
                                                .WORD   Save R2,R3,R4                                   ; 2073
                54 00000000'    EF   9E 00002           MOVAB   QA$AW_OVERALL_ERROR_TABLE, R4           ;
                53 00000000G    9F   9E 00009           MOVAB   @#DS$PRINTF, R3                         ;
                52 00000000'    EF   9E 00010           MOVAB   HEADER_LINE_1, R2                       ;
                01 00000000G    EF   D1 00017           CMPL    DS$GL_FSTTEST, #1                       ; 2155
                                78   12 0001E           BNEQ    1$                                      ;
                50 00000254     9F   D0 00020           MOVL    @#^X00000254, R0                        ;
                60 00000000G    EF   D1 00027           CMPL    DS$GL_LSTTEST, (R0)                     ;
                                78   12 0002E           BNEQ    2$                                      ;
                                52   DD 00030           PUSHL   R2                                      ; 2162
                63              01   FB 00032           CALLS   #1, DS$PRINTF                           ; 2163
                        29      A2   9F 00035           PUSHAB  HEADER_LINE_2                           ; 2164
                63              01   FB 00038           CALLS   #1, DS$PRINTF                           ; 2165
                        4C      A2   9F 0003B           PUSHAB  HEADER_LINE_3                           ; 2177
                63              01   FB 0003E           CALLS   #1, DS$PRINTF                           ; 2179
                        75      A2   9F 00041           PUSHAB  HEADER_LINE_4                           ; 2187
                63              01   FB 00044           CALLS   #1, DS$PRINTF                           ; 2189
                7E              64   3C 00047           MOVZWL  QA$AW_OVERALL_ERROR_TABLE, -(SP)        ; 2197
                        0087    C2   9F 0004A           PUSHAB  NS_LINE                                 ; 2199
                63              02   FB 0004E           CALLS   #2, DS$PRINTF                           ; 2207
                                64   B5 00051           TSTW    QA$AW_OVERALL_ERROR_TABLE               ; 2209
                                62   12 00053           BNEQ    3$                                      ; 2216
                7E              02   A4 3C 00055        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+2, -(SP)
                7E 00000000G    EF   3C 00059           MOVZWL  QA$W_MULTIPLEPASS, -(SP)
                        00A4    C2   9F 00060           PUSHAB  MP_LINE
                63              03   FB 00064           CALLS   #3, DS$PRINTF
                        02      A4   B5 00067           TSTW    QA$AW_OVERALL_ERROR_TABLE+2
                                48   12 0006A           BNEQ    3$
                7E              04   A4 3C 0006C        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+4, -(SP)
                7E 00000000G    EF   3C 00070           MOVZWL  QA$W_TESTLOOPS, -(SP)
                        00CF    C2   9F 00077           PUSHAB  LT_LINE
                63              03   FB 0007B           CALLS   #3, DS$PRINTF
                        04      A4   B5 0007E           TSTW    QA$AW_OVERALL_ERROR_TABLE+4
                                34   12 00081           BNEQ    3$
                7E              06   A4 3C 00083        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+6, -(SP)
                7E 00000000G    EF   3C 00087           MOVZWL  QA$W_SUBTESTLOOPS, -(SP)
                        0102    C2   9F 0008E           PUSHAB  LS_LINE
                63              03   FB 00092           CALLS   #3, DS$PRINTF
                        06      A4   B5 00095           TSTW    QA$AW_OVERALL_ERROR_TABLE+6
                                1D   12 00098 1$:        BNEQ    3$
                7E              08   A4 3C 0009A        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+8, -(SP)
                        0138    C2   9F 0009E           PUSHAB  RB_LINE
                63              02   FB 000A2           CALLS   #2, DS$PRINTF
```

```
                                      08   A4  B5 000A5          TSTW    QA$AW_OVERALL_ERROR_TABLE+8                     ; 2218
                                           0D  12 000A8 2$:     BNEQ    3$                                              ;
                                00000208   9F  DD 000AA          PUSHL   @#^X00000208                                    ; 2221
                                    015C   C2  9F 000B0          PUSHAB  OK_LINE                                         ;
                                63         02  FB 000B4          CALLS   #2, DS$PRINTF                                   ;
                                           04  000B7 3$:         RET                                                     ; 2224

; Routine Size:  184 bytes,    Routine Base:  CODE + 057B
```

```
;   2225  1     %SBTTL 'End of Module QAMAIN'
;   2226  1
;   2227  0     END ELUDOM
```

```
;                                  PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  DATA                          1159  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                            53  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          1587  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                               Library Statistics
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265              784        23         2          85       00:00.1
;  DRB1:[DS.WORK]DS.L32;159                653        11         1          42       00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7         18017        12         0         975       00:04.7
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE QAMAIN

; Size:          1587 code + 1212 data bytes
; Run Time:         00:45.4
; Elapsed Time:     02:26.6
; Lines/CPU Min:     2945
; Lexemes/CPU-Min: 19395
; Memory Used:  297 pages
; Compilation Complete
```

Table of contents

(1)	106	Libraries and Equated Symbols
(1)	242	External Symbols
(1)	340	EXE$QIO QIO DISPATCHING ROUTINE
(2)	389	QIO$INITIALIZE Initialize QIO database
(4)	446	QIO$CLEANUP Remove old QIO database
(5)	545	QIO$ADDUNIT Add a unit to the QIO database
(7)	803	INI_MBADP BUILD ADP AND INITIALIZE MBA
(9)	900	INI_UBADP BUILD ADP AND INITIALIZE UBA
(10)	957	INI_DRADP BUILD ADP AND INITIALIZE DRA
(12)	1054	MAXIMIZE ACCESS MODE
(13)	1080	BUG$CHECK Bugcheck notification routine
(14)	1115	QIO$LOAD_DB Insert unit into data base
(16)	1397	DB_ERROR ERROR LOADING DATABASE
(16)	1469	I/O DATABASE INTERLOCKS
(17)	1477	IOGEN$CNTRL_INI Initialize a controller
(18)	1531	MEMORY ALLOCATION SUBROUTINES
(19)	1573	ADPLINK Link adapter to end of ADP list
(19)	1607	Exe$PwrTimChk - Check reasonable interval since power recovery ;[14]

```
0000     1          .TITLE  QIO
0000     2          .IDENT  /07-18/
0000     3          .NoShow Conditionals
0000     4          .DSABL  GBL
0000     5
0000     6  ;
0000     7  ; Copyright (c) 1977, 1982, 1984
0000     8  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     9  ;
0000    10  ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    11  ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    12  ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    13  ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    14  ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    15  ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    16  ; REMAIN IN DEC.
0000    17  ;
0000    18  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    19  ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    20  ; CORPORATION.
0000    21  ;
0000    22  ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    23  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    24
0000    25  ;++
0000    26  ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    27  ;
0000    28  ; ABSTRACT:
0000    29  ;
0000    30  ; ENVIRONMENT:
0000    31  ;
0000    32  ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    33  ; MODIFIED BY:
0000    34  ;                N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    35  ;       01       QIO SUPPORT
0000    36  ;                TOM SOUTTER     22-FEB-78
0000    37  ;       02       ADDED CALL TO VMS/QIO HANDLER
0000    38  ;                R. RIGGS        1-OCT-78        VERSION 3 (ESSAA-5.00)
0000    39  ;       03       INCLUDED QIO DATABASE INITIALIZATION ROUTINES
0000    40  ;                N.Howgate       26-Jun-79       Version 4 (ESSAA-5.01)
0000    41  ;       04       Add support for 11/750
0000    42  ;                Roger Riggs     03-NOV-1979
0000    43  ;       05       Added code to load drivers if not found in
0000    44  ;                IOC$GL_DPTLIST
0000    45  ;       06       Added fudge to generate 2 vectors for LPA11K
0000    46  ;                Roger Riggs, 5-Feb-1980, Version 5.2
0000    47  ;       07       Revamped setup of ACF block, Added code to maintain reference
0000    48  ;                count in DPT headers, added code to handle headers on drivers
0000    49  ;                header contains the number of vectors for unibus  devices
0000    50  ;                Removes need for fudge in edit 06.
0000    51  ;--
0000    52  ;
0000    53  ;       08       John Ciukaj     5-May-80        Version 5.4
0000    54  ;
0000    55  ;                  Added Dummy return for QIO related calls
0000    56  ;
0000    57  ;                Dave Butenhof   18-jun-1980,    version 5.5
```

```
0000     58 ;        09      Remove all dummy returns from this module, and move to
0000     59 ;                new module VMSDUMMY.
0000     60 ;                Roger Riggs, 10-Aug-1980
0000     61 ;        10      Use QIO driver prefix from controller instead of device
0000     62 ;                this allows for terminals to be connected to more than
0000     63 ;                one type of Async controller.  The first prefix
0000     64 ;                encountered while linking back through the P-tables is
0000     65 ;                used.
0000     66 ;
0000     67 ;        11      - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     68 ;                Fixed truncation errors. Also added .LIBRARY statements
0000     69 ;                for $DS and $DIAG. Also changed .PSECT SEP to WORK, CODE,
0000     70 ;                and DATA where appropriate.
0000     71 ;
C000     72 ;        12      - Jack Stansbury, 12-Nov-1981, Version 6.-
0000     73 ;                Changed the HALTD to HALT.
0000     74 ;
0000     75 ;        13      Marion Baggett, 9-Apr-1982, Version 6.7
0000     76 ;                Added .NoShow Conditionals after .IDENT statement.
0000     77 ;
0000     78 ;        14      Jack Stansbury, July 1, 1982, Version 6.8
0000     79 ;                Added the 'real' Exe$PwrTimChk routine.
0000     80 ;
0000     81 ;        15      Marion Baggett, 24-Sept-1982, Version 6.9
0000     82 ;                Changed the CASE instruction since WRITEVBLK and READVBLK
0000     83 ;                are to far away and a JMP was need.
0000     84 ;
0000     85 ;        16      Richard E. Muse, 9 January, 1984, Version 6.14
0000     86 ;                Added $PRDEF macro definition - VMS Version 4 change.
0000     87 ;
0000     88 ;        17      Domenic Andella, 9-Apr-1984, Version 6.14
0000     89 ;                Added code in FILL_ACF to handle case where
0000     90 ;                an SBIA is attached, and it is necessary to
0000     91 ;                'backtrack' another level.
0000     92 ;
0000     93 ;        18      Bob Bergazzi    24-Apr-1984     Version 7.0
0000     94 ;                        - Changed FILL_ACF to set up R3 with the CSR
0000     95 ;                          address for the subsequent call to FIND_ADP.
0000     96 ;                        - Changed FIND_ADP routine to compare ADP$L_CSR
0000     97 ;                          instead of ADP$W_TR. This accomodates the xxx
0000     98 ;                          cpu with multiple SBIs.
0000     99 ;                        - Changed INI_DRADP to add 100 or 300 to SCB_BASE
0000    100 ;                          depending on which SBI the adapter is connected to.
0000    101 ;                        - Removed UBINT... code from INI_UBADP since it
0000    102 ;                          only needs the size of that code (UBINTSZ), which
0000    103 ;                          is defined in ONLYXXX.
0000    104 ;-
```

```
                        .Sbttl  Libraries and Equated Symbols

;
; INCLUDE FILES:
;
                        .Library        /SYS$LIBRARY:LIB/
                        .LIBRARY        /$DS/                   ;                               [11]
                        .LIBRARY        /$DIAG/                 ;                               [11]

;
; MACROS:
;
                        .MACRO  $PRDEF,$GBL                                     ; [16]

                        $DEFINI PR,$GBL

                $EQU    PR$_KSP 0
                $EQU    PR$_ESP 1
                $EQU    PR$_SSP 2
                $EQU    PR$_USP 3
                $EQU    PR$_ISP 4
                $EQU    PR$_P0BR        8
                $EQU    PR$_P0LR        9
                $EQU    PR$_P1BR        10
                $EQU    PR$_P1LR        11
                $EQU    PR$_SBR 12
                $EQU    PR$_SLR 13
                $EQU    PR$_PCBB        16
                $EQU    PR$_SCBB        17
                $EQU    PR$_IPL 18
                $EQU    PR$_ASTLVL      19
                $EQU    PR$_SIRR        20
                $EQU    PR$_SISR        21
                $EQU    PR$_ICCS        24
                $EQU    PR$_NICR        25
                $EQU    PR$_ICR 26
                $EQU    PR$_TODR        27
                $EQU    PR$_RXCS        32
                $EQU    PR$_RXDB        33
                $EQU    PR$_TXCS        34
                $EQU    PR$_TXDB        35
                $EQU    PR$_ACCS        40
                $EQU    PR$_ACCR        41
                $EQU    PR$_MAPEN       56
                $EQU    PR$_TBIA        57
                $EQU    PR$_TBIS        58
                $EQU    PR$_PME 61
                $EQU    PR$_SID 62
                $EQU    PR$_TBCHK       63
                $EQU    PR$V_SID_SN     0
                $EQU    PR$S_SID_SN     12
                $EQU    PR$V_SID_PL     12
                $EQU    PR$S_SID_PL     3
                $EQU    PR$V_SID_ECO    15
                $EQU    PR$S_SID_ECO    9
                $EQU    PR$V_SID_TYPE   24
                $EQU    PR$S_SID_TYPE   8
```

```
0000   163  $EQU   PR$_SID_TYP780  1
0000   164  $EQU   PR$_SID_TYP750  2
0000   165  $EQU   PR$_SID_TYP730  3
0000   166  $EQU   PR$_SID_TYP7VV  4
0000   167  $EQU   PR$_SID_TYPMAX  4
0000   168  $EQU   PR$_WCSA        44
0000   169  $EQU   PR$_WCSD        45
0000   170  $EQU   PR$_SBIFS       48
0000   171  $EQU   PR$_SBIS        49
0000   172  $EQU   PR$_SBISC       50
0000   173  $EQU   PR$_SBIMT       51
0000   174  $EQU   PR$_SBIER       52
0000   175  $EQU   PR$_SBITA       53
0000   176  $EQU   PR$_SBIQC       54
0000   177  $EQU   PR$_CMIERR      23
0000   178  $EQU   PR$_CSRS        28
0000   179  $EQU   PR$_CSRD        29
0000   180  $EQU   PR$_CSTS        30
0000   181  $EQU   PR$_CSTD        31
0000   182  $EQU   PR$_TBDR        36
0000   183  $EQU   PR$_CADR        37
0000   184  $EQU   PR$_MCESR       38
0000   185  $EQU   PR$_CAER        39
0000   186  $EQU   PR$_UBRESET     55
0000   187  $EQU   PR$_PAMACC      64
0000   188  $EQU   PR$_PAMLOC      65
0000   189  $EQU   PR$_CSWP        66
0000   190  $EQU   PR$_CRBT        67
0000   191  $EQU   PR$_MCTL1       68
0000   192  $EQU   PR$_MCTL2       69
0000   193  $EQU   PR$_MGEN        70
0000   194  $EQU   PR$_MTBER       71
0000   195  $EQU   PR$_MEAR        72
0000   196  $EQU   PR$_MDCR1       73
0000   197  $EQU   PR$_MEDR        74
0000   198  $EQU   PR$_MECCR       75
0000   199
0000   200        $DEFEND PR,$GBL,DEF
0000   201
0000   202        .ENDM   $PRDEF
0000   203        $PRDEF                                  ; CALL PRDEF MACRO
0000   204
0000   205 ;
0000   206 ; EQUATED SYMBOLS:
0000   207 ;
0000   208        .NLIST  MEB
0000   209        $ACFDEF
0000   210        $ADPDEF
0000   211        $ALLOCDEF
0000   212        $CCBDEF
0000   213        $CRBDEF
0000   214        $DCDEF
0000   215        $DDBDEF
0000   216        $DDTDEF
0000   217        $DPTDEF
0000   218        $DYNDEF GLOBAL
0000   219        $IDBDEF
```

```
0000    220          $IODEF
0000    221          $PSLDEF
0000    222          $QIODEF
0000    223          $SYSGMSGDEF
0000    224          $UBADEF
0000    225          $UCBDEF
0000    226          $VECDEF
0000    227          $DS_TYPEDEF
0000    228          CMKDEF
0000    229
0000    230          $DS_QIODVR_DEF                  ; Offsets in driver header
0000    231          $DS_HPDDEF
0000    232          $DS_RH780_DEF
0000    233          $DS_DW780_DEF
0000    234          $DS_DR780_DEF
0000    235          $DS_SCBDEF
0000    236          $DS_LOAD_DEF
0000    237          $DS_DW750_DEF
0000    238          $DS_DSDEF
0000    239          $DS_DSADEF
0000    240          .LIST MEB
```

```
                                0000   242          .Subtitle       External Symbols
                                0000   243
                                0000   244          .EXTRN  IOC$GL_ADPLIST, IOC$GL_DEVLIST, IOC$GL_DPTLIST
                                0000   245          .EXTRN  EXE$ALONONPAGED,EXE$DEANONPAGED
                                0000   246          .EXTRN  IO$TESTCSR_L,   IO$TESTCSR_W
                                0000   247          .EXTRN  DSX$PRINT,      DS_ERRSUP,      DS$GPHARD
                                0000   248          .EXTRN  DS$LOAD,        DSR$COMPLETION
                                0000   249          .EXTRN  MBA$INITIAL,    MBA$INT,        UBA$UNEXINT
                                0000   250          .EXTRN  DR$INITIAL,     DR$INT
                                0000   251          .EXTRN  UBA$INITIAL,    UBINTSZ                         ;[18.
                                0000   252          .EXTRN  SCB_BASE,       SCB_IMAGE,      UBADP_CPU
                                0000   253          .EXTRN  DS$R_CCB,       VMS$QIO,        IOGEN$CONN_VEC
                                0000   254          .EXTRN  WRITEVBLK,      READVBLK,       PREADVBLK
                                0000   255          .EXTRN  SS$_INSFMEM,    SS$_NORMAL,     SS$_IVCHAN
                                0000   256          .EXTRN  IOC$INITDRV,    SCAN$SEARCHLIST,SCAN$ALPHA
                                0000   257          .EXTRN  IOC$IOPOST,     EXE$FORKDSPTH,  IOC$REINITDRV
                                0000   258          .EXTRN  LOC$PTDESC,     QIOCLEAN_CPU
                                0000   259          .EXTRN  IOC$RTN
                                0000   260
```

```
                                00000000   262          .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                                [11]
                                    0000   263
                                    0000   264          MODNAM  QIO
                     4F 49 51 00'   0000       $MODULE:          .ASCIC  \QIO\
                                    0004   265
                                    0004   266 EXE$GL_PWRDONE::                          ; End time for power up interval        [14]
                          00000000  0004   267          .LONG   0                        ; Done now                              [14]
                                    0008   268
                                    0008   269 AUNKFUNC:
20 6E 77 6F 6E 6B 6E 55 20 3F 3F 00' 0008   270          .ASCIC  \?? Unknown QIO function or channel number zero.\
6E 6F 69 74 63 6E 75 66 20 4F 49 51  0014
20 6C 65 6E 6E 61 68 63 20 72 6F 20  0020
2E 6F 72 65 7A 20 72 65 62 6D 75 6E  002C
                              2F     0008
                                    0038   271
                                    0038   272          .ALIGN  LONG
                                    0038   273 CTL$GW_NMIOCH::          ; NUMBER OF CHANNELS
                              FFFF' 0038   274          .WORD   DS$K_CCB - 1
                                    003A   275
                                    003A   276 CTL$GW_CHINDX::          ; MAXIMUM CHANNEL INDEX + 1
                              0000' 003A   277          .WORD   DS$K_CCB @ 4
                                    003C   278
                                    003C   279 MBINT_DISP:                              ;INTERRUPT DISPATCHER MODEL
                           3C   BB  003C   280          PUSHR   #^M<R2,R3,R4,R5>
                 00000000  9F   16  003E   281          JSB     @#0
                                    0044   282
                                    0044   283 UBINT_DISP::
                           3F   BB  0044   284          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
                                    0046   285
                                    0046   286 T_NODRIVER:
21 20 52 45 56 49 52 44 20 3F 3F 00' 0046   287          .ASCIC  "?? DRIVER !AS NOT FOUND FOR !AS!/"
44 4E 55 4F 46 20 54 4F 4E 20 53 41  0052
      2F 21 53 41 21 20 52 4F 46 20  005E
                              21     0046
                                    0068   288
                                    0068   289 T_WRONGVER:
49 4D 20 54 4E 45 44 49 20 3F 3F 00' 0068   290          .ASCIC  "?? IDENT MISMATCH, !AS=!UW, SUPERVISOR=!UW!/"
3D 53 41 21 20 2C 48 43 54 41 4D 53  0074
49 56 52 45 50 55 53 20 2C 57 55 21  C080
         2F 21 57 55 21 3D 52 4F 53  008C
                              2C     0068
                                    0095   291
                                    0095   292 T_INVADP:
49 20 53 49 20 43 41 21 20 3F 3F 00' 0095   293          .ASCIC  "?? !AC IS INCORRECT ADAPTER TYPE FOR !AS!/"
41 44 41 20 54 43 45 52 52 4F 43 4E  00A1
4F 46 20 45 50 59 54 20 52 45 54 50  00AD
               2F 21 53 41 21 20 52  0089
                              2A     0095
                                    00C0   294
                                    00C0   295 ;
                                    00C0   296 ; Table to convert adapter name to adapter type, Parallel to T_KNOWN_DEV
                                    00C0   297 ;
                                    00C0   298 T_ADAPTER_TYPE:
                              01    00C0   299          .BYTE   AT$_UBA                  ; DW780
                              00    00C1   300          .BYTE   AT$_MBA                  ; RH780
                              01    00C2   301          .BYTE   AT$_UBA                  ; DW750
```

```
                        00  00C3    302          .BYTE   AT$_MBA                 ; RH750
                        01  00C4    303          .BYTE   AT$_UBA                 ; DW730
                        02  00C5    304          .BYTE   AT$_DR                  ; RH780
                            00C6    305
                            00C6    306 T_KNOWN_DEV:
                    0006'   00C6    307          .WORD   $_KNOWN
                    000C'   00C8    308          .WORD   10$-.
                    0010'   00CA    309          .WORD   20$-.
                    0014'   00CC    310          .WORD   30$-.
                    0018'   00CE    311          .WORD   40$-.
                    001C'   00D0    312          .WORD   50$-.
                    0020'   00D2    313          .WORD   60$-.
                00000006    00D4    314 $_KNOWN=<.-T_KNOWN_DEV-1>/2
      30 38 37 57 41 00'    00D4    315 10$:     .ASCIC  "DW780"
                     05     00D4
      30 38 37 48 52 00'    00DA    316 20$:     .ASCIC  "RH780"
                     05     00DA
      30 35 37 57 44 00'    00E0    317 30$:     .ASCIC  "DW750"
                     05     00E0
      30 35 37 48 52 00'    00E6    318 40$:     .ASCIC  "RH750"
                     05     00E6
      30 33 37 57 44 00'    00EC    319 50$:     .ASCIC  "DW730"
                     05     00EC
      30 38 37 52 44 00'    00F2    320 60$:     .ASCIC  "DR780"
                     05     00F2
```

```
                    00000000   322          .Psect  Work, NoExe, NoShr, Wrt, Long   ;                                    [11]
                        0000   323 QIO$L_ALLOCATE:
                    00000000   0000   324          .LONG                           ; BYTES ALLOCATED TO QIO DATA STRUCTURES
                        0004   325
                        0004   326 LOAD_FLAGS:                                    ;CONTROL BLOCKS CREATED FLAGS
                    00000005   0004   327          .BLKB   1
                        0005   328          _VIELD  LOAD,0,<-                       ; FLAG DEFINTION'
                        0005   329                  <DDB,,M>,-                      ;  DDB CREATED
                        0005   330                  <CRB,,M>,-                      ;  CRB CREATED
                        0005   331                  <IDB,,M>,-                      ;  IDB CREATED
                        0005   332                  <UCB,,M>,-                      ;  UCB CREATED
                        0005   333                  >
                        0005   334 DDB_BLINK:                                     ;DDB BACKWARD LINK
                    00000009   0005   335          .BLKL   1
                        C009   336
                        0009   337 UCB_BLINK:                                     ;UCB BACKWARD LINK
                    0000000D   0009   338          .BLKL   1
```

```
                        000D   340          .SBTTL  EXE$QIO          QIO DISPATCHING ROUTINE
                    00000000   341          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                [11]
                        0000   342 ;++
                        0000   343 ; FUNCTIONAL DESCRIPTION:
                        0000   344 ;
                        0000   345 ;
                        0000   346 ; CALLING SEQUENCE:
                        0000   347 ;
                        0000   348 ;       NONE
                        0000   349 ;
                        0000   350 ; INPUT PARAMETERS:     NONE
                        0000   351 ;
                        0000   352 ; IMPLICIT INPUTS:      NONE
                        0000   353 ;
                        C000   354 ; OUTPUT PARAMETERS:    NONE
                        0000   355 ;
                        0000   356 ; IMPLICIT OUTPUTS:     NONE
                        0000   357 ;
                        0000   358 ; COMPLETION CODES:     NONE
                        0000   359 ;
                        0000   360 ; SIDE EFFECTS:         NONE
                        0000   361 ;
                        0000   362 ;--
                    0FDC 0000  363 .ENTRY  EXE$QIO,^M<R2,R3,R4,R6,R7,R8,R9,R10,R11>
                        0002   364
                 08 AC  B5 0002  365          TSTW    QIO$_CHAN(AP)           ; LOOK AT CHANNEL NUMBER
                    08  13 0005  366          BEQL    5$                      ; IF 0, DO 'DS' CONSOLE I/O
     00000000'EF    6C  FA 0007  367          CALLG   (AP),L^VMS$QIO                  ; Else call VMS/QIO handler     [11]
                    05  000E  368          RSB                             ;  AND RETURN THRU VECTOR
                        000F   369                                          ;
 50  0C AC  06  00  EF  000F  370 5$:       EXTZV   #IO$V_FCODE,#IO$S_FCODE, -
                        0015   371                  QIO$_FUNC(AP),R0        ; Get function code
                        0015   372          CASE    R0,TYPE=L,LIMIT=#IO$_WRITEVBLK,DISPLIST=<-
                        0015   373                  30$, -                  ; Function ^X30
                        0015   374                  40$, -                  ; Function ^X31
                        0015   375                  20$, -                  ; Function ^X32
                        0015   376                  20$, -                  ; Function ^X33
                        0015   377                  20$, -                  ; Function ^X34
                        0015   378                  20$, -                  ; Function ^X35
                        0015   379                  20$, -                  ; Function ^X36
                        0015   380                  40$>                    ; Function ^X37
        07'  30  50  CF 0015          CASEL   R0,#IO$_WRITEVBLK,S^#<<30001$-30000$>/2>-1
                        0019       30000$:
                0026'   0019               .SIGNED_WORD    30$-30000$
                002C'   001B               .SIGNED_WORD    40$-30000$
                0010'   001D               .SIGNED_WORD    20$-30000$
                0010'   001F               .SIGNED_WORD    20$-30000$
                0010'   0021               .SIGNED_WORD    20$-30000$
                0010'   0023               .SIGNED_WORD    20$-30000$
                0010'   0025               .SIGNED_WORD    20$-30000$
                002C'   0027               .SIGNED_WORD    40$-30000$
                        0029       30001$:
                        0029   381
                        0029   382 20$:     ERRSUP_S                MSGADR=AUNKFUNC
          00000000'EF   DF 0029                       PUSHAL  $MODULE
          00000008'EF   DF 002F                       PUSHAL  AUNKFUNC
                    01  DD 0035                       PUSHL   #$ER
```

```
                00000000'EF   03   FB  0037                          CALLS   #3, DS_ERRSUP
                                   04  003E   383                RET
                     00000000'EF   17  003F   384 30$:           JMP     WRITEVBLK               ; This address was out of bounds for    [15]
                                       0045   385                                               ; the case instruction                  [15]
                     00000000'EF   17  0045   386 40$:           JMP     READVBLK                ; This address was out of bounds for    [15]
                                       0048   387                                               ; the case instruction                  [15]
```

```
                                004B   389          .SBTTL  QIO$INITIALIZE  Initialize QIO database
                                004B   390 ;++
                                004B   391 ; FUNCTIONAL DESCRIPTION:
                                004B   392 ;
                                004B   393 ;       This routine will initialize all of the the database that needs it.
                                004B   394 ;       It resets the DRIVER prologue queue.
                                004B   395 ;
                                004B   396 ; CALLING SEQUENCE:
                                004B   397 ;
                                004B   398 ;       BSBW    QIO$INITIALIZE
                                004B   399 ;
                                004B   400 ; INPUT PARAMETERS:      NONE
                                004B   401 ;
                                004B   402 ; IMPLICIT INPUTS:       NONE
                                C04B   403 ;
                                004B   404 ; OUTPUT PARAMETERS:     NONE
                                004B   405 ;
                                004B   406 ; IMPLICIT OUTPUTS:      NONE
                                004B   407 ;
                                004B   408 ; COMPLETION CODES:      NONE
                                004B   409 ;
                                004B   410 ; SIDE EFFECTS:          NONE
                                C04B   411 ;--
                                004B   412
                          0000  004B   413 .ENTRY  QIO$INITIALIZE,^M<>
                                004D   414
    51   00000000'EF    DE     004D   415          MOVAL   L^IOC$GL_DPTLIST,R1     ; GET ADDRESS OF DRIVER QUEUE LISTHEAD  [11]
         61    51       D0     0054   416          MOVL    R1,(R1)                 ; INIT FLINK
       04 A1   51       D0     0057   417          MOVL    R1,4(R1)                ; INIT BLINK
                                005B   418
         00000000'EF    D4     005B   419          CLRL    L^IOC$GL_DEVLIST        ; CLEAR DDB HEADER LIST                 [11]
         00000000'EF    D4     0061   420          CLRL    L^IOC$GL_ADPLIST        ; CLEAR ADAPTER HEADER LIST             [11]
         00000000'EF    D4     0067   421          CLRL    L^QIO$L_ALLOCATE        ; CLEAR QIO MEMORY ALLOCATED            [11]
                                006D   422
                          04    006D   423          RET                             ; RETURN
```

```
                                006E   425 ;+
                                006E   426 ; Find DPT given driver name descriptor in R2,R3
                                006E   427 ;-
                                006E   428 LOC$DPT:
                     007C 8F    BB  006E   429          PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save descriptor of driver name
                58 00000000'EF  9E  0072   430          MOVAB   L^IOC$GL_DPTLIST,R11    ; Address of DPT list head                [11]
                        56 5B   D0  0079   431          MOVL    R11,R6                  ; Save list header
                                    007C   432
                        5B 6B   D0  007C   433 10$:     MOVL    DPT$L_FLINK(R11),R11    ; Link to next DPT
                     50 5B 56   C3  007F   434          SUBL3   R6,R11,R0               ; End of list?, R0 = ZERO if so
                           15   13  0083   435          BEQL    20$                     ; Branch if end
                                    0085   436
                        52 6E   7D  0085   437          MOVQ    (SP),R2                 ; Get descriptor of driver name
                     51 1E AB   9E  0088   438          MOVAB   DPT$T_NAME(R11),R1      ; Get address of driver name
                        50 81   9A  008C   439          MOVZBL  (R1)+,R0                ; Get length
             63 52 00   61 50   2D  008F   440          CMPC5   R0,(R1),#0,R2,(R3)      ; Same?
                           E5   12  0095   441          BNEQ    10$                     ; Branch if not
                        50 C0'  D0  0097   442          MOVL    S^#SS$_NORMAL,R0        ; Success
                     007C 8F    BA  009A   443 20$:     POPR    #^M<R2,R3,R4,R5,R6>     ; Restore
                                05  009E   444          RSB                             ; Return R11 has DPT if R0=SS$_NORMAL
```

```
                                009F    446          .SBTTL  QIO$CLEANUP     Remove old QIO database
                                009F    447 ;++
                                009F    448 ; FUNCTIONAL DESCRIPTION:
                                009F    449 ;
                                009F    450 ;       This routine is used to remove and release all of the
                                009F    451 ;       QIO data base.
                                009F    452 ;       Every ADP is deallocated while restoring the SCB for each.
                                009F    453 ;       Scan thru the DDB list, making a list on the stack of the CRB's
                                009F    454 ;       present and deallocating each UCB and then the DDB.
                                009F    455 ;       Finally deallocate the space for each CRB and corresponding IDB.
                                009F    456 ;
                                009F    457 ; CALLING SEQUENCE:
                                009F    458 ;
                                009F    459 ;       CALLG   (SP),QIO$CLEANUP
                                009F    460 ;
                                009F    461 ; INPUT PARAMETERS:     NONE
                                009F    462 ;
                                009F    463 ; IMPLICIT INPUTS:      NONE
                                009F    464 ;
                                009F    465 ; OUTPUT PARAMETERS:    NONE
                                009F    466 ;
                                009F    467 ; IMPLICIT OUTPUTS:     NONE
                                009F    468 ;
                                009F    469 ; COMPLETION CODES:     NONE
                                009F    470 ;
                                009F    471 ; SIDE EFFECTS:
                                009F    472 ;
                                009F    473 ;       The current QIO database is destroyed.
                                009F    474 ;--
                                009F    475
                        007C    009F    476 .ENTRY  QIO$CLEANUP,^M<R2,R3,R4,R5,R6>
                                00A1    477
 55   00000000'EF       DE      00A1    478         MOVAL   SCB_BASE,R5             ; BASE SCB
 54   00000000'EF       DE      00A8    479         MOVAL   SCB_IMAGE,R4            ; BASE SCB IMAGE
                                00AF    480         DSBINT                          ; Don't allow interruptions
            7E    12    DB      00AF                        MFPR    S^#PR$_IPL,-(SP)
            12    1F    DA      00B2                        MTPR    #31,S^#PR$_IPL
                                00B5    481
 53   00000000'EF       DE      00B5    482         MOVAL   L^IOC$GL_ADPLIST,R3     ; POINT TO ADAPTER HEADER                [11]
            50    63    D0      00BC    483 10$:    MOVL    (R3),R0                 ; POINT TO FIRST ADAPTER BLOCK
      00000000'EF       16      00BF    484         JSB     QIOCLEAN_CPU                    ; Do CPU specific QIO cleanup    [11]
00A0 C5   00A0 C4       D0      00C5    485         MOVL    SCB$L_SFTLVL8(R4), -
                                00CC    486                 SCB$L_SFTLVL8(R5)       ; RETURN SOFT VECTOR OF FORK DISPATCH
0090 C5   0090 C4       D0      00CC    487         MOVL    SCB$L_SFTLVL4(R4), -
                                00D3    488                 SCB$L_SFTLVL4(R5)       ; RETURN SOFT VECTOR OF IOPOST
```

```
                    7E   D4  00D3   490          CLRL    -(SP)                          ; TERMINATOR FOR CRB LIST
                                00D5   491
          56  00000000'EF   D0  00D5   492 30$:    MOVL    L^IOC$GL_DEVLIST,R6            ; GET FIRST DDB                    [11]
                    3F   13  00DC   493          BEQL    100$                           ; BRANCH IF NONE
                                00DE   494
                50  04 A6   D0  00DE   495          MOVL    DDB$L_UCB(R6),R0               ; POINT TO FIRST UCB
                55  20 A0   D0  00E2   496          MOVL    UCB$L_CRB(R0),R5               ; GET CRB ADDRESS
                                00E6   497 40$:
                   51  6E   9E  00E6   498          MOVAB   (SP),R1                        ; POINT TO FIRST ELIGIBLE
                       20   BB  00E9   499          PUSHR   #^M<R5>                        ; ASSUME NOT A DUPLICATE
                       61   D5  00EB   500 50$:     TSTL    (R1)                           ; VALID
                       07   13  00ED   501          BEQL    60$                            ; BRANCH IF AT END OF LIST
                   55  81   D1  00EF   502          CMPL    (R1)+,R5                       ; SAME?
                       F7   12  00F2   503          BNEQ    50$                            ; BRANCH IF NOT THE SAME
                       8E   D5  00F4   504          TSTL    (SP)+                          ; REMOVE DUPLICATE CRB
                                00F6   505 60$:
                55  10 A5   D0  00F6   506          MOVL    CRB$L_LINK(R5),R5              ; LINK TO NEXT CRB
                       EA   12  00FA   507          BNEQ    40$                            ; CHECK IT
                                00FC   508 70$:
                55  D8 A6   DE  00FC   509          MOVAL   DDB$L_UCB-UCB$L_LINK(R6),R5 ; FIRST UCB
                55  2C A5   D0  0100   510 80$:     MOVL    UCB$L_LINK(R5),R5              ; LINK TO NEXT UCB
                       08   13  0104   511          BEQL    90$                            ; BRANCH IF NONE
                   50  55   D0  0106   512          MOVL    R5,R0                          ; COPY ADDRESS
                     08D4   30  0109   513          BSBW    QIO$DEALLOCATE                 ; RELEASE THE SPACE
                       F2   11  010C   514          BRB     80$                            ; NEXT UCB
                                010E   515 90$:
                50  56   D0  010E   516          MOVL    R6,R0                          ; COPY DDB ADDRESS
00000000'EF  66   D0  0111   517          MOVL    DDB$L_LINK(R6),L^IOC$GL_DEVLIST ;      [11]
                     08C5   30  0118   518          BSBW    QIO$DEALLOCATE                 ; RELEASE THE DDB
                       B8   11  011B   519          BRB     30$                            ; NEXT DDB
                                011D   520 100$:
                56  8E   D0  011D   521          MOVL    (SP)+,R6                       ; GET CRB FROM STACK
                       0F   13  0120   522          BEQL    110$                           ; BRANCH IF LAST ONE
             50  1C A6   D0  0122   523          MOVL    CRB$L_INTD+VEC$L_IDB(R6),R0 ; POINT TO IDB
                     08B7   30  0126   524          BSBW    QIO$DEALLOCATE                 ; RELEASE THE IDB
                50  56   D0  0129   525          MOVL    R6,R0                          ; COPY CRB ADDRESS
                     08B1   30  012C   526          BSBW    QIO$DEALLOCATE                 ; RELEASE THE CRB
                       EC   11  012F   527          BRB     100$                           ; CHECK FOR MORE CRB'S
                                0131   528 110$:
                                0131   529
51  00000000'EF   DE  0131   530          MOVAL   L^IOC$GL_DPTLIST,R1    ; Address of header                [11]
                50  51   D0  0138   531          MOVL    R1,R0                          ; Duplicate
                50  60   D0  013B   532 130$:     MOVL    DPT$L_FLINK(R0),R0             ; Link to next
                51  50   D1  013E   533          CMPL    R0,R1                          ; Point back to header?
                       05   13  0141   534          BEQL    140$                           ; Branch if back to header
                     08 A0   94  0143   535          CLRB    DPT$B_REFC(R0)                 ; Clear reference count
                       F3   11  0146   536          BRB     130$                           ; next DPT
                                0148   537 140$:
                                0148   538          ENBINT                                 ; Enable interrupts again
                                                            MTPR    (SP)+,S^#PR$_IPL
                12   8E   DA  0148
52  00000000'EF   D0  014B   539          MOVL    QIO$L_ALLOCATE,R2              ; SHOULD ALL BE DEALLOCATED
                       11   13  0152   540          BEQL    120$                           ; BRANCH IF IT WAS
                                0154   541          ERRSUP_S
                00000000'EF   DF  0154                       PUSHAL  $MODULE
                       00   DD  015A                       PUSHL   #0
                       02   DD  015C                       PUSHL   #$ER
00000000'EF   03   FB  015E                       CALLS   #3, DS_ERRSUP
```

```
                    0165   542 120$:
                04  0165   543          RET
```

```
                        .SBTTL  QIO$ADDUNIT     Add a unit to the QIO database
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine takes all the necessary steps to
                ;       insert a unit into the QIO database, including adding
                ;       ADP's, CRB's, IDB's, DDB's and UCB's.
                ;       The generic name of the device is used to locate the
                ;       device driver for the device.  Starting with the specified
                ;       device fields are copied from the P-table to the ACF block
                ;       which contains all the information necessary to build the
                ;       QIO database.  For unknown devices the unit and vector information
                ;       is copied. For a TM03 the unit is copied to only one of two fields.
                ;       For channels, (DR780, RH780 & DW780) the configuration register is set
                ;       and the ACF is complete. For all non-channel devices this process
                ;       is repeated for the link device, and then for it's link device...etc.
                ;       Note that it is impossible to catch the case where the TM03 is
                ;       omitted from the link chain for a magtape drive... i.e. no
                ;       error is reported.  Errors are reported if the channel is omitted.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLG/S arglist/count,QIO$ADDUNIT
                ;
                ; INPUT PARAMETERS:
                ;
                ;       0(AP)   #1
                ;       4(AP)   Address of P-table
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       Existing QIO database
                ;
                ; OUTPUT PARAMETERS:    NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       The QIO database may be updated.
                ;
                ; COMPLETION CODE:
                ;
                ;       SS$_NORMAL      if succcssful
                ;
                ; REGISTER USAGE:
                ;
                ;       R3      TR number of adapter
                ;       R4      Adapter CSR address
                ;       R5      Address of Driver Prologue Table
                ;       R6      Address of ADP
                ;       AP      Address of current P-table
                ;       R7      Address of ACF block
                ;       R10     Address of Adapter P-table
                ;       R11     Address of Controller P-table
                ;       AP      Address of Unit P-table
                ;
                ; SIDE EFFECTS:         NONE
                ;--
```

```
                        0166   602          $OFFSET 0,NEGATIVE,< -
                        0166   603          <FILE,12>, -                    ; File name of driver
                        0166   604          <FILEDESC,8>, -                 ; Descriptor of driver file name
                        0166   605          <FILEDEF,8>, -                  ; Default descriptor
                        0166   606          <LENGTH,4>, -                   ; Longword to recieve file length
                        0166   607          <ACF,ACF$C_LENGTH>, -           ; Place to build an ACF
                        0166   608          <LOCAL,0>>
                        0166                .SAVE   LOCAL_BLOCK
                        0166                .PSECT $ABS$ ABS
                00000000  FE70              .=0
                FFFFFFF4  0000              .BLKB   -12
                          FFF4        FILE:
                          FFEC        FILEDESC:
                          FFE4        FILEDEF:
                          FFE0        LENGTH:
                          FFC0        ACF:
                          FFC0        LOCAL:
```

```
                         0FFC  0166   610 .ENTRY  QIO$ADDUNIT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                               0168   611
      5E    C0 AD    9E      0168   612          MOVAB   LOCAL(FP),SP              ; Allocate stack based storage
      57    C0 AD    9E      016C   613          MOVAB   ACF(FP),R7               ; Base ACF
50    00000000'EF    DE      0170   614          MOVAL   SCB_BASE,R0              ; Base SCB
                               0177   615
      5C    04 AC    D0      0177   616          MOVL    4(AP),AP                ; Address P-table
                               017B   617
55    04 AC    01    C1      017B   618          ADDL3   #1,HP$Q_DEVICE+4(AP),R5 ; Point past "_"
   54    6C    01    C3      0180   619          SUBL3   #1,HP$Q_DEVICE(AP),R4   ; Length of device name
      00000000'EF    16      0184   620          JSB     SCAN$ALPHA              ; Determine length of alphabetic
         52    50    D0      018A   621          MOVL    R0,R2                   ; Save length of name
         51    50    C0      018D   622          ADDL    R0,R1                   ; Point past string
         7E    71    90      0190   623 10$:     MOVB    -(R1),-(SP)             ; Copy device name onto stack
            FA 50    F5      0193   624          SOBGTR  R0,10$                  ; Count them all
         7E    52    90      0196   625          MOVB    R2,-(SP)                ; Make it ASCIC
                               0199   626
      14 A7    6E    9E      0199   627          MOVAB   (SP),ACF$L_DEVNAME(R7)  ; Set address of device name
                               019D   628
         56    5C    D0      019D   629          MOVL    AP,R6                   ; Copy P-table address
      51    26 A6    9E      01A0   630 20$:     MOVAB   HP$T_TYPE(R6),R1        ; Address of device type
         50    81    9A      01A4   631          MOVZBL  (R1)+,R0                ; Get length of device type
      00000000'EF    16      01A7   632          JSB     LOC$PTDESC              ; Locate this PT-descriptor
                13    13      01AD   633          BEQL    40$                     ; Branch to NODRIVER
                               01AF   634
         51    80    9A      01AF   635          MOVZBL  (R0)+,R1                ; Get length of type name
   50    02 A041    9E      01B2   636          MOVAB   2(R0)[R1],R0            ; Get address of driver prefix
         52    60    3C      01B7   637          MOVZWL  (R0),R2                 ; Get prefix
               0E    12      01BA   638          BNEQ    50$                     ; Branch if valid prefix
      56    20 A6    D0      01BC   639          MOVL    HP$A_LINK(R6),R6        ; Link to previous device
               DE    12      01C0   640          BNEQ    20$                     ; Check for prefix in controller
50    006600B1 8F    D0      01C2   641 40$:     MOVL    #DS$_NOSUPPORT,R0       ; No QIO support for device
                     04      01C9   642          RET                             ; Exit
                               01CA   643
                               01CA   644 ;+
                               01CA   645 ; Locate driver, given generic device prefix in R2
                               01CA   646 ;-
50    00000000'EF    7E      01CA   647 50$:     MOVAQ   L^IOC$GL_DPTLIST,R0     ; Point to driver listhead                [11]
         55    50    D0      01D1   648          MOVL    R0,R5                   ; Copy listhead
         55    65    D0      01D4   649 60$:     MOVL    DPT$L_FLINK(R5),R5      ; Link to next driver
         55    50    D1      01D7   650          CMPL    R0,R5                   ; At end?
               09    13      01DA   651 70$:     BEQL    LOAD_DRIVER             ; Branch if more drivers to check
      1F A5    52    B1      01DC   652          CMPW    R2,DPT$T_NAME+1(R5)     ; Match driver prefix
               F2    12      01E0   653          BNEQ    60$                     ; Try again no match
                  00CC   31      01E2   654          BRW     FILL_ACF                ; Branch since driver found
```

```
                                01E5    656 ;+
                                01E5    657 ; Load device driver
                                01E5    658 ; Inputs:       R2 - 2 character generic device name
                                01E5    659 ; Outputs:      R4 - Address of DPT
                                01E5    660 ;-
                                01E5    661 LOAD_DRIVER:
      F4 AD    3F515645 8F    D0  01E5    662         MOVL    #^A"EVQ?",FILE(FP)          ; Set first part of name
               F7 AD    52    B0  01ED    663         MOVW    R2,FILE+3(FP)               ; Insert generic device name
      F9 AD    4558452E 8F    D0  01F1    664         MOVL    #^A".EXE",FILE+5(FP)        ; Read .EXE file
               EC AD    09    D0  01F9    665         MOVL    #9,FILEDESC(FP)             ; Set length of file spec
            F0 AD    F4 AD    9E  01FD    666         MOVAB   FILE(FP),FILEDESC+4(FP)     ; Set address of file spec
                     E4 AD    7C  0202    667         CLRQ    FILEDEF(FP)                 ; Clear default descriptor
                                0205    668
                     E0 AD    DF  0205    669         PUSHAL  LENGTH(FP)                  ; Address to return length of file
                        7E    7C  0208    670         CLRQ    -(SP)                       ; Length of buffer, and address
                     E4 AD    7F  020A    671         PUSHAQ  FILEDEF(FP)                 ; Address of default
                     EC AD    7F  020D    672         PUSHAQ  FILEDESC(FP)                ; Address of file spec
                        05    DD  0210    673         PUSHL   #5                          ; Length of arg list
         00000000'EF    6E    FA  0212    674         CALLG   (SP),DS$LOAD                ; Locate existence and length of file
                     1D 50    E8  0219    675         BLBS    R0,50$                      ; Branch if file found
                        01    BB  021C    676         PUSHR   #^M<R0>                     ; Save status code
                        6C    9F  021E    677         PUSHAB  HP$Q_DEVICE(AP)             ; Name of device
                     EC AD    9F  0220    678         PUSHAB  FILEDESC(FP)                ; descriptor of file name
                  00000046'EF    9F  0223    679         PUSHAB  L^T_NODRIVER                ; Address of edit string                [11]
                        7E    01  9A  0229    680         movzbl  #ds$k_printf,-(sp)          ; use PRINTF                            [13]
                        7E    22  98  022C    681         cvtbl   #ds$k_type_qio_nodriver,-(sp)   ; code                          [13]
         00000000'GF    05    FB  022F    682         CALLS   #5, G^DSX$PRINT             ; Print the error                       [13]
                        01    BA  0236    683         POPR    #^M<R0>                     ; Restore failing DS$LOAD status
                              04  0238    684         RET                                 ; Exit
                                0239    685
               51    E0 AD    D0  0239    686 50$:    MOVL    LENGTH(FP),R1               ; Length of memory required
            00000000'EF    16  023D    687         JSB     EXE$ALONONPAGED             ; Allocate the memory for the driver
                  07 50    E8  0243    688         BLBS    R0,55$                      ; Continue if OK
            00000000'EF    16  0246    689         JSB     DSR$COMPLETION              ; Print the error
                              04  024C    690         RET                                 ; Return
                                024D    691
                  53    51    3C  024D    692 55$:    MOVZWL  R1,R3                       ; Save length of file
               0C AE    51    3C  0250    693         MOVZWL  R1,LOAD$_LENGTH(SP)         ; Store length to read
               10 AE    52    D0  0254    694         MOVL    R2,LOAD$_ADDRESS(SP)        ; Store address of buffer
         00000000'EF    8E    FB  0258    695         CALLS   (SP)+,DS$LOAD               ; Now read the file
                  08 50    E8  025F    696         BLBS    R0,60$                      ; Continue if OK
            00000000'EF    16  0262    697         JSB     DSR$COMPLETION              ; Print the error
                        2D  11  0268    698         BRB     70$                         ; Error return, release driver space
                                026A    699
               08 A2    53    B0  026A    700 60$:    MOVW    R3,DVR$W_SIZE(R2)           ; Stash size of driver
      10 A2    FFFF0001 8F    D1  026E    701         CMPL    #DVR$_IDENT,-
                                0276    702                 DVR$L_IDENT(R2)             ; Is ident version same as supervisor?
                        2B  13  0276    703         BEQL    80$                         ; Branch if so
                  FFFF0001 8F    DD  0278    704         PUSHL   #DVR$_IDENT                 ; Correct ident
                        10 A2    DD  027E    705         PUSHL   DVR$L_IDENT(R2)             ; Current ident
                     EC AD    7F  0281    706         PUSHAQ  FILEDESC(FP)                ; Address of driver
                  00000068'EF    9F  0284    707         PUSHAB  L^T_WRONGVER                ; Edit string                           [11]
                        7E    01  9A  028A    708         movzbl  #ds$k_printf,-(sp)          ; Use PRINTF                            [13]
                        7E    23  98  028D    709         cvtbl   #ds$k_type_qio_wrongver,-(sp)   ; code                          [13]
         00000000'GF    06    FB  0290    710         CALLS   #6, G^DSX$PRINT             ; Print the error                       [13]
                                0297    711 ;
                                0297    712 ; Release the driver and return error
```

```
                                0297   713 ;
                          50    52  D0  0297   714 70$:    MOVL    R2,R0                       ; Point to buffer
                00000000'EF     16  029A   715            JSB     EXE$DEANONPAGED             ; Release the space
                          50    D4  02A0   716            CLRL    R0                          ; Error return
                                04  02A2   717            RET                                 ; Exit
                                    02A3   718
                       55 14 A2   9E  02A3   719 80$:    MOVAB   DVR$C_LEN(R2),R5            ; Base driver prologue table
                          08 A5   B4  02A7   720            CLRW    DPT$W_SIZE(R5)              ; Clear size of DPT, catch bugs later
            00000000'EF    65     0E  02AA   721            INSQUE  (R5),IOC$GL_DPTLIST         ; Insert in queue
```

```
                        02B1    723 ;+
                        02B1    724 ; Fill ACF block
                        02B1    725 ;-
                        02B1    726 FILL_ACF:
      18 A7   1E A5   9E 02B1    727         MOVAB   DPT$T_NAME(R5),-
                        02B6    728                 ACF$L_DRVNAME(R7)       ; Address of driver name
                        02B6    729
                        02B6    730
         5B   20 AC   D0 02B6    731         MOVL    HP$A_LINK(AP),R11       ; Address of Controller P-TABLE
                  03   12 02BA    732         BNEQ    20$                     ; Branch if real controller
            5B    5C   D0 02BC    733         MOVL    AP,R11                  ; Use unit for controller
         5A   20 AB   D0 02BF    734 20$:    MOVL    HP$A_LINK(R11),R10      ; Address of Adapter P-TABLE
                  06   12 02C3    735         BNEQ    40$                     ; Branch if real adapter
            5A    5B   D0 02C5    736         MOVL    R11,R10                 ; Use controller for adapter
            5B    5C   D0 02C8    737         MOVL    AP,R11                  ; Use unit for controller
                        02CB    738
41494253 8F   27 AA   D1 02CB    739 40$:    CMPL    HP$T_TYPE+1(R10),#^A'SBIA' ; SBIA attached?                     ;[17
                  06   12 02D3    740         BNEQ    45$                     ; Branch if not                         ;[17
            5A    5B   D0 02D5    741         MOVL    R11,R10                 ; backtrack through ptables             ;[17
            5B    5C   D0 02D8    742         MOVL    AP,R11                  ; ...                                   ;[17
                        02DB    743
         51   26 AA   9E 02DB    744 45$:    MOVAB   HP$T_TYPE(R10),R1       ; Address of adapter type               ;[17
            50   81   9A 02DF    745         MOVZBL  (R1)+,R0                ; Length of adapter type
      53 000000C6'EF   9E 02E2    746         MOVAB   L^T_KNOWN_DEV,R3        ; Point to list of known types          [11]
         00000000'EF   16 02E9    747         JSB     SCAN$SEARCHLIST         ; Locate which type it is
                  29   12 02EF    748         BNEQ    60$                     ; Branch if unknown adapter..error
   52 000000C0'EF40   9A 02F1    749         MOVZBL  T_ADAPTER_TYPE[R0],R2   ; Get adapter type
         0C A5   52   91 02F9    750         CMPB    R2,DPT$B_ADPTYPE(R5)    ; Correct type of adapter?
                  1B   13 02FD    751         BEQL    60$                     ; Branch if correct
                  6C   9F 02FF    752 50$:    PUSHAB  HP$Q_DEVICE(AP)         ; Address of device name descriptor
                  26 AA   9F 0301    753         PUSHAB  HP$T_TYPE(R10)          ; Address of adapter type
            00000095'EF   9F 0304    754         PUSHAB  T_INVADP                ; Address of text
                  7E   01   9A 030A    755         movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                            [13]
                  7E   24   98 030D    756         cvtbl   #ds$k_type_qio_invadp,-(sp)     ; code                  [13]
   00000000'GF    05   FB 0310    757         CALLS   #5, G^DSX$PRINT         ; Print the error                       [13]
                  50   D4 0317    758         CLRL    R0                      ; Error return
                  04   9A 0319    759         RET
                        031A    760
         54   18 AA   D0 031A    761 60$:    MOVL    HP$A_DEVICE(R10),R4     ; Get adapter CSR address
   53    54   04   0D   EF 031E    762         EXTZV   #13,#4,R4,R3            ; Get TR number or SLOT-16
                  06E7   30 0323    763         BSBW    FIND_ADP                ; Locate adapter control block
                  1E 50   E8 0326    764         BLBS    R0,70$                  ; Branch if found
                  47'AF   9F 0329    765         PUSHAB  B^70$                   ; Dummy BSBW return
                        032C    766         CASE    R2,LIMIT=#AT$_MBA,TYPE=L,DISPLIST=-
                        032C    767                 <INI_MBADP,INI_UBADP,INI_DRADP>
                        0336    768         ERRSUP_S
                        0347    769
            67   56   D0 0347    770 70$:    MOVL    R6,ACF$L_ADAPTER(R7)    ; Set ADP control block address
      04 A7    54   D0 034A    771         MOVL    R4,ACF$L_CONFIGREG(R7)  ; Set adapter csr address
   08 A7   24 AA   B0 034E    772         MOVW    HP$W_VECTOR(R10),-
                        0353    773                 ACF$Q_AVECTOR(R7)       ; Set adapter vector
   0A A7   0B AB   90 0353    774         MOVB    HP$B_DRIVE(R11),-
                        0358    775                 ACF$B_AUNIT(R7)         ; Set adapter unit
   0C A7   18 AB   D0 0358    776         MOVL    HP$A_DEVICE(R11),-
                        035D    777                 ACF$L_CONTRLREG(R7)     ; Set device csr register
   10 A7   24 AB   B0 035D    778         MOVW    HP$W_VECTOR(R11),-
                        0362    779                 ACF$Q_CVECTOR(R7)       ; Set Controller vector
```

```
                        05   12  0362   780          BNEQ    80$                     ; Branch if real vector
                10 A7   24 AA   B0  0364   781          MOVW    HP$W_VECTOR(R10), -
                                    0369   782                  ACF$W_CVECTOR(R7)   ; Set Controller vector
                12 A7   0B AC   90  0369   783 80$:     MOVB    HP$B_DRIVE(AP), -
                                    036E   784                  ACF$B_CUNIT(R7)     ; Set controller unit
                1C A7   16 A5   B0  036E   785          MOVW    DPT$W_MAXUNITS(R5), -
                                    0373   786                  ACF$W_MAXUNITS(R7)  ; Set max number of units
                   50   EC A5   9E  0373   787          MOVAB   -DVR$C_LEN(R5),R0   ; Base of DVR header
                13 A7   0B A0   90  0377   788          MOVB    DVR$B_CNUMVEC(R0), -
                                    037C   789                  ACF$B_CNUMVEC(R7)   ; Set number of controller vectors
                        03   12  037C   790          BNEQ    90$                     ; Branch if really one there
                        13 A7   96  037E   791          INCB    ACF$B_CNUMVEC(R7)   ; Default is one
                        0B A7   94  0381   792 90$:     CLRB    ACF$B_AFLAG(R7)     ; Clear adapter flag
                                    0384   793
        00000638'EF     67   FA  0384   794          CALLG   (R7),L^QIO$LOAD_DB      ; Load the database                    [11]
                    0B A5   96  038B   795          INCB    DPT$B_REFC(R5)          ; Flag as successful
                    19 50   E8  038E   796          BLBS    R0,999$                 ; Branch if it worked
                    0B A5   97  0391   797          DECB    DPT$B_REFC(R5)          ; It failed, decrement the count
                        50   DD  0394   798          PUSHL   R0                      ; Save return code
                                    0396   799          ERRSUP_S                    ; Spaz
                        50 8ED0  03A7   800          POPL    R0                      ; Restore reason
                             04  03AA   801 999$:     RET
```

```
                        03AB    803          .SBTTL  INI_MBADP       BUILD ADP AND INITIALIZE MBA
                        03AB    804 ;+
                        03AB    805 ; INI_MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                        03AB    806 ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                        03AB    807 ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                        03AB    808 ; BY CALLING MBA$INITIAL.
                        03AB    809 ;
                        03AB    810 ; INPUT:
                        03AB    811 ;       R3 - TR NUMBER OF MASSBUS ADAPTER
                        03AB    812 ;       R4 - CONFIGURATION REGISTER OF ADAPTER
                        03AB    813 ; OUTPUT:
                        03AB    814 ;       R6 - ADP ADDRESS
                        03AB    815 ;
                        03AB    816 ;-
                        03AB    817
                        03AB    818 INI_MBADP:
    07BC 8F    BB   03AB    819          PUSHR   #^M<R2,R3,R4,R5,R7,R8,R9,R10>
    57    53   D0   03AF    820          MOVL    R3,R7                       ; COPY TR NUMBER
                   03B2    821
    51    38   9A   03B2    822          MOVZBL  #CRB$C_LENGTH,R1            ; SET SIZE OF CRB
       0605    30   03B5    823          BSBW    QIO$ALLOCATE                ; ALLOCATE SPACE FOR CRB
      36 50    E9   03B8    824          BLBC    R0,70$                      ; BRANCH IF ALLOCATION FAILED
    5A    52   D0   03BB    825          MOVL    R2,R10                      ; SAVE CRB ADDRESS
08 AA    51   B0   03BE    826          MOVW    R1,CRB$W_SIZE(R10)          ; SET DATA STRUCTURE SIZE
                   03C2    827
51    54 8F    9A   03C2    828          MOVZBL  #IDB$C_LENGTH+<8*4>,R1      ; SET SIZE TO ALLOCATE FOR IDB
       05F4    30   03C6    829          BSBW    QIO$ALLOCATE                ; ALLOCATE SPACE FOR IDB
      1F 50    E9   03C9    830          BLBC    R0,60$                      ; BRANCH IF ALLOCATION FAILED
    59    52   D0   03CC    831          MOVL    R2,R9                       ; COPY ADDRESS OF IDB
08 A9    51   B0   03CF    832          MOVW    R1,IDB$W_SIZE(R9)           ; SET DATA STRUCTURE SIZE
                   03D3    833
    51    14   9A   03D3    834          MOVZBL  #ADP$C_MBAADPLEN,R1         ; SET SIZE TO ALLOCATE FOR ADP
       05E4    30   03D6    835          BSBW    QIO$ALLOCATE                ; ALLOCATE SPACE FOR ADP
      09 50    E9   03D9    836          BLBC    R0,50$                      ; BRANCH IF ALLOCATION FAILED
    56    52   D0   03DC    837          MOVL    R2,R6                       ; ADDRESS OF ADP
08 A6    51   B0   03DF    838          MOVW    R1,ADP$W_SIZE(R6)           ; SET DATA STRUCTURE SIZE
          14    11   03E3    839          BRB     100$                        ; ALL ALLOCATED FINISH INITIALIZATION
                   03E5    840
                   03E5    841
    50    59   D0   03E5    842 50$:     MOVL    R9,R0                       ; COPY ADDRESS
       05F5    30   03E8    843          BSBW    QIO$DEA_LOCATE              ; RELEASE
    50    5A   D0   03EB    844 60$:     MOVL    R10,R0                      ; COPY ADDRESS OF CRB
       05EF    30   03EE    845          BSBW    QIO$DEALLOCATE              ; RELEASE
                   03F1    846 70$:
50  0000'8F    3C   03F1    847          MOVZWL  #SS$_INSFMEM,R0             ; RETURN CODE
       0091    31   03F6    848          BRW     999$                        ; EXIT
```

```
                                03F9    850 100$:
                                03F9    851         DSBINT                              ; Disable interrupts
                          3C    BB  03FF    852         PUSHR   #^M<R2,R3,R4,R5>            ; SAVE MOVC REGISTERS
     14 AA   0000048F'EF  10'   28  0401    853         MOVC3   S^#MBACRBLEN,L^MBACRB,CRB$L_INTD(R10) ; FILL WITH TEMPLATE      [11]
                          3C    BA  040A    854         POPR    #^M<R2,R3,R4,R5>            ; RESTORE MOVC REGISTERS
                                040C    855
                                040C    856 ; Add offset of X100 or X300 based on SBIA number gleaned from R4.                  ;[17
                                040C    857                                                                                     ;[17
                    0A 54   19  E0  040C    858         BBS     #25,R4,110$                 ; CHECK WHICH SBIA 0 OR 1           ;[17
           50   00000100'EF47  DE  0410    859         MOVAL   SCB_BASE+^X100[R7],R0       ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA0);[1
                          08    11  0418    860         BRB     120$                        ; CONTINUE                          ;[17
           50   00000300'EF47  DE  041A    861 110$:   MOVAL   SCB_BASE+^X300[R7],R0       ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA1);[1
                    60   15 AA  DE  0422    862 120$:   MOVAL   CRB$L_INTD+1(R10),(R0)      ; CONNECT VECTOR TO CRB CODE        ;[17
                 40 A0   15 AA  DE  0426    863         MOVAL   CRB$L_INTD+1(R10),64(R0)    ; SAME FOR
              0080 C0    15 AA  DE  042B    864         MOVAL   CRB$L_INTD+1(R10),128(R0)   ; ALL FOUR
              00C0 C0    15 AA  DE  0431    865         MOVAL   CRB$L_INTD+1(R10),192(R0)   ; VECTORS
                    0A AA   05  90  0437    866         MOVB    #DYN$C_CRB,CRB$B_TYPE(R10)  ; SET CORRECT TYPE
                       6A   6A  DE  043B    867         MOVAL   CRB$L_WQFL(R10),CRB$L_WQFL(R10) ; INITIALIZE WAIT QUEUE HEADER
                    04 AA   6A  DE  043E    868         MOVAL   CRB$L_WQFL(R10),CRB$L_WQBL(R10) ; FLINK AND BLINK
                    0C AA   08  B0  0442    869         MOVW    #8,CRB$W_REFC(R10)          ; SET REFERENCE COUNT
                                0446    870
                    0A A9   09  90  0446    871         MOVB    #DYN$C_IDB,IDB$B_TYPE(R9)   ; AND TYPE CODE
                    0C A9   08  9B  044A    872         MOVZBW  #8,IDB$W_UNITS(R9)          ; Set count of units
                       69   54  D0  044E    873         MOVL    R4,IDB$L_CSR(R9)            ; SET CSR ADDRESS TO CONFIG REG 0
                    1C AA   59  D0  0451    874         MOVL    R9,CRB$L_INTD+VEC$L_IDB(R10)    ; SET ADDRESS OF IDB INTO CRB
           20 AA   00000000'9F  9E  0455    875         MOVAB   @#MBA$INITIAL,CRB$L_INTD+VEC$L_INITIAL(R10) ; SET ADDRESS OF CONTROL
                                045D    876
                    0A A6   01  90  045D    877         MOVB    #DYN$C_ADP,ADP$B_TYPE(R6)   ; TYPE CODE
                       66   54  D0  0461    878         MOVL    R4,ADP$L_CSR(R6)            ; SET ADDRESS OF CONFIGURATION REGISTER
                    0C A6   57  B0  0464    879         MOVW    R7,ADP$W_TR(R6)             ; SET TR NUMBER OF ADAPTER
                    0E A6   00  B0  0468    880         MOVW    #AT$_MBA,ADP$W_ADPTYPE(R6)  ; SET ADAPTER TYPE TO MBA
                    10 A6   5A  D0  046C    881         MOVL    R10,ADP$L_CRB(R6)           ; POINT ADP TO CRB
                    28 AA   56  D0  0470    882         MOVL    R6,CRB$L_INTD+VEC$L_ADP(R10)    ; SET CRB POINTER TO ADP
                    10 A9   56  D0  0474    883         MOVL    R6,IDB$L_ADP(R9)            ; AND INTO IDB
                       55   59  D0  0478    884         MOVL    R9,R5                       ; ADDRESS OF IDB
                         0578  30  047B    885         BSBW    ADPLINK                     ; LINK ADP TO END OF CHAIN
                00000000'EF  16  047E    886         JSB     MBA$INITIAL                 ; INITIALIZE MASSBUS ADAPTER
                                0484    887         ENBINT                              ; Re-enable interrupts
                       50   00'  D0  0487    888         MOVL    S^#SS$_NORMAL,R0            ; SUCCESSFUL
                                048A    889
                    07BC 8F     BA  048A    890 999$:   POPR    #^M<R2,R3,R4,R5,R7,R8,R9,R10>; RESTORE ALL REGISTERS
                                05  048E    891         RSB                                 ; RETURN
                                048F    892
                                048F    893 MBACRB:                                     ; SKELETON MBA CRB
                          3C    BB  048F    894         PUSHR   #^M<R2,R3,R4,R5>            ; SAVE REGISTERS
                00000000'9F     16  0491    895         JSB     @#MBA$INT                   ; CALL INTERRUPT DISPATCHER
                         00000000   0497    896         .LONG   0                           ; CRB$L_INTD+VEC$L_INTD
                         00000000'  049B    897         .LONG   MBA$INITIAL                 ; CRB$L_INTD+VEC$L_INITIAL
                         00000010   049F    898 MBACRBLEN=.-MBACRB                          ;
```

```
                                049F   900          .SBTTL  INI_UBADP       BUILD ADP AND INITIALIZE UBA
                                049F   901 ;+
                                049F   902 ; INI_UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
                                049F   903 ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
                                049F   904 ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
                                049F   905 ;
                                049F   906 ; INPUT:
                                049F   907 ;       R3 - TR NUMBER
                                049F   908 ;       R4 - CONFIGURATION REGISTER OF ADAPTER
                                049F   909 ; OUTPUT:
                                049F   910 ;       R6 - ADP ADDRESS
                                049F   911 ; REGISTERS:
                                049F   912 ;       R4 - ADAPTER CONFIGURATION REGISTER ADDR
                                049F   913 ;       R6 - ADP ADDRESS
                                049F   914 ;       R7 - TR NUMBER
                                049F   915 ;-
                    00000080    049F   916          NUMUBAVEC = 128                 ; ALLOW FOR 128 UNIBUS VECTORS
                                049F   917
                                049F   918 INI_UBADP:
                00BC 8F   BB    049F   919          PUSHR   #^M<R2,R3,R4,R5,R7>     ; SAVE R2-R7
                57   53   D0    04A3   920          MOVL    R3,R7                   ; COPY TR NUMBER
              51  0270'8F   3C    04A6   921          MOVZWL  #<ADP$C_UBAADPLEN+UBINTSZ+<NUMUBAVEC*4>>,R1
                                04AB   922                                          ; SET SIZE OF BLOCK TO ALLOCATE
                  050F   30    04AB   923          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR ADP
                03 50    E8    04AE   924          BLBS    R0,10$                  ; EXIT IF ERROR
                  0062   31    04B1   925          BRW     60$                     ; EXIT ON ERROR
                                04B4   926 10$:
                                04B4   927          DSBINT                          ; Disable interrupts
                   56   52    D0    04BA   928          MOVL    R2,R6                   ; COPY ADP ADDRESS
              08 A6   51    B0    04BD   929          MOVW    R1,ADP$W_SIZE(R6)       ; SET SIZE INTO ADP BLOCK
              0A A6   01    90    04C1   930          MOVB    #DYN$C_ADP,ADP$B_TYPE(R6); AND SET TYPE OF BLOCK
              0E A6   01    B0    04C5   931          MOVW    #AT$_UBA,ADP$W_ADPTYPE(R6) ; SET TYPE OF ADAPTER
                   66   54    D0    04C9   932          MOVL    R4,ADP$L_CSR(R6)        ; SET VA OF CONFIGURATION REG
              0C A6   57    B0    04CC   933          MOVW    R7,ADP$W_TR(R6)         ; SET TR NUMBER FOR ADAPTER
              50    14 A6   DE    04D0   934          MOVAL   ADP$L_DPQFL(R6),R0      ; ADDRESS OF DATA PATH WAIT QUEUE
                   60   50    D0    04D4   935          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
              04 A0   50    D0    04D7   936          MOVL    R0,4(R0)                ;
              50    1C A6   DE    04DB   937          MOVAL   ADP$L_MRQFL(R6),R0      ; ADDRESS OF MAP WAIT QUEUE
                   60   50    D0    04DF   938          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
              04 A0   50    D0    04E2   939          MOVL    R0,4(R0)                ;
                   50   1E    D0    04E6   940          MOVL    #30,R0                  ; NUMBER OF WORDS-1 IN MAP BITMAP
           26 A640    01    AE    04E9   941 20$:     MNEGW   #1,ADP$W_MRBITMAP(R6)[R0] ; SET 16 BITS OF BITMAP
                 F8 50    F4    04EE   942          SOBGEQ  R0,20$                  ; FILL 31 * 16  => 496 BITS
                 04 A6    D4    04F1   943          CLRL    ADP$L_LINK(R6)          ; ZAP ADAPTER CHAIN LINK
                  FB09'   30    04F4   944          BSBW    UBA$INITIAL             ; AND INITIALIZE ADAPTER
                  04FC    30    04F7   945          BSBW    ADPLINK                 ; LINK ADP TO END OF LIST
                  FB03'   30    04FA   946          BSBW    UBADP_CPU               ; CALL CPU DEPENDANT CODE
              50    10 A6   D0    04FD   947          MOVL    ADP$L_VECTOR(R6),R0     ; GET ADDRESS OF VECTOR
           51    0080 8F   3C    0501   948          MOVZWL  #NUMUBAVEC,R1           ; SET NUMBER OF VECTORS
        80  00000000'EF   9E    0506   949 50$:     MOVAB   UBA$UNEXINT,(R0)+       ; FILL VECTOR WITH UNEXPECTED INTERRUPT
                 F6 51    F5    050D   950          SOBGTR  R1,50$                  ; DO ALL VECTOR POSITIONS
                           0510   951          ENBINT                          ; Enable interrupts
                   50   01    D0    0513   952          MOVL    #1,R0                   ; SUCCESS
                           0516   953 60$:
                00BC 8F   BA    0516   954          POPR    #^M<R2,R3,R4,R5,R7>     ; RESTORE R2-R7
                           05    051A   955          RSB                             ; AND RETURN
```

```
                                0518   957          .SBTTL  INI_DRADP       BUILD ADP AND INITIALIZE DRA
                                0518   958 ;+
                                0518   959 ; INI_DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                                0518   960 ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                                0518   961 ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                                0518   962 ; BY CALLING DRA$INITIAL.
                                0518   963 ;
                                0518   964 ; INPUT:
                                0518   965 ;       R3 - TR NUMBER OF MASSBUS ADAPTER
                                0518   966 ;       R4 - CONFIGURATION REGISTER OF ADAPTER
                                0518   967 ; OUTPUT:
                                0518   968 ;       R6 - ADP ADDRESS
                                0518   969 ;
                                0518   970 ;-
                                0518   971
                                0518   972 INI_DRADP:
                    07BC 8F     BB  0518   973          PUSHR   #^M<R2,R3,R4,R5,R7,R8,R9,R10>
                    57   53     D0  051F   974          MOVL    R3,R7                   ; COPY TR NUMBER
                                    0522   975
                    51   38     9A  0522   976          MOVZBL  #CRB$C_LENGTH,R1        ; SET SIZE OF CRB
                       0495     30  0525   977          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR CRB
                      35 50     E9  0528   978          BLBC    R0,70$                  ; BRANCH IF ALLOCATION FAILED
                    5A   52     D0  052B   979          MOVL    R2,R10                  ; SAVE CRB ADDRESS
                08  AA   51     B0  052E   980          MOVW    R1,CRB$W_SIZE(R10)      ; SET DATA STRUCTURE SIZE
                                    0532   981
                    51   34     9A  0532   982          MOVZBL  #IDB$C_LENGTH,R1        ; SET SIZE TO ALLOCATE FOR IDB
                       0485     30  0535   983          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR IDB
                      1F 50     E9  0538   984          BLBC    R0,60$                  ; BRANCH IF ALLOCATION FAILED
                    59   52     D0  053B   985          MOVL    R2,R9                   ; COPY ADDRESS OF IDB
                08  A9   51     B0  053E   986          MOVW    R1,IDB$W_SIZE(R9)       ; SET DATA STRUCTURE SIZE
                                    0542   987
                    51   14     9A  0542   988          MOVZBL  #ADP$C_DRADPLEN,R1      ; SET SIZE TO ALLOCATE FOR ADP
                       0475     30  0545   989          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR ADP
                      09 50     E9  0548   990          BLBC    R0,50$                  ; BRANCH IF ALLOCATION FAILED
                    56   52     D0  054B   991          MOVL    R2,R6                   ; ADDRESS OF ADP
                08  A6   51     B0  054E   992          MOVW    R1,ADP$W_SIZE(R6)       ; SET DATA STRUCTURE SIZE
                          14     11  0552   993          BRB     100$                    ; ALL ALLOCATED FINISH INITIALIZATION
                                    0554   994
                                    0554   995
                    50   59     D0  0554   996 50$:     MOVL    R9,R0                   ; COPY ADDRESS
                       0486     30  0557   997          BSBW    QIO$DEALLOCATE          ; RELEASE
                    50   5A     D0  055A   998 60$:     MOVL    R10,R0                  ; COPY ADDRESS OF CRB
                       0480     30  055D   999          BSBW    QIO$DEALLOCATE          ; RELEASE
                                    0560  1000 70$:
                50  0000'8F     3C  0560  1001          MOVZWL  #SS$_INSFMEM,R0         ; RETURN CODE
                          8F     31  0565  1002          BRW     999$                    ; EXIT
```

```
                                0568  1004 100$:
                                0568  1005          DSBINT                                  ; Disable interrupts
                          3C    056E  1006          PUSHR   #^M<R2,R3,R4,R5>                ; SAVE MOVC REGISTERS
          14 AA   05FC'CF 10'   0570  1007          MOVC3   S^#DRACRBLEN,W^DRACRB,CRB$L_INTD(R10) ; FILL WITH TEMPLATE
                          3C    0577  1008          POPR    #^M<R2,R3,R4,R5>                ; RESTORE MOVC REGISTERS
                                0579  1009
                                0579  1010 ; Add offset of X100 or X300 based on SBIA number gleaned from R4.                [18]
                                0579  1011
                       0A 54    19   057A  1012          BBS     #25,R4,110$                     ; CHECK WHICH SBIA 0 OR 1                [18]
               50  00000100'EF47   057D  1013          MOVAL   SCB_BASE+^X100[R7],R0           ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA0)[18
                          08    0585  1014          BRB     120$                            ; CONTINUE                               [18]
               50  00000300'EF47   0587  1015 110$:   MOVAL   SCB_BASE+^X300[R7],R0           ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA1)[18
                    60    15 AA   058F  1016 120$:   MOVAL   CRB$L_INTD+1(R10),(R0)          ; CONNECT VECTOR TO CRB CODE
                 40 A0    15 AA   0593  1017          MOVAL   CRB$L_INTD+1(R10),64(R0)        ; SAME FOR
              0080 C0    15 AA   0598  1018          MOVAL   CRB$L_INTD+1(R10),128(R0)       ; ALL FOUR
              00C0 C0    15 AA   059E  1019          MOVAL   CRB$L_INTD+1(R10),192(R0)       ; VECTORS
                    0A AA   05   05A4  1020          MOVB    #DYN$C_CRB,CRB$B_TYPE(R10)      ; SET CORRECT TYPE
                       6A   6A   05A8  1021          MOVAL   CRB$L_WQFL(R10),CRB$L_WQFL(R10) ; INITIALIZE WAIT QUEUE HEADER
                    04 AA   6A   05AB  1022          MOVAL   CRB$L_WQFL(R10),CRB$L_WQBL(R10) ; FLINK AND BLINK
                    0C AA   08   05AF  1023          MOVW    #8,CRB$W_REFC(R10)              ; SET REFERENCE COUNT
                                0583  1024
                    0A A9   09   05B3  1025          MOVB    #DYN$C_IDB,IDB$B_TYPE(R9)       ; AND TYPE CODE
                    0C A9   01   05B7  1026          MOVZBW  #1,IDB$W_UNITS(R9)              ; Set count of units
                       69   54   05B8  1027          MOVL    R4,IDB$L_CSR(R9)                ; SET CSR ADDRESS TO CONFIG REG 0
                    1C AA   59   05BE  1028          MOVL    R9,CRB$L_INTD+VEC$L_IDB(R10)    ; SET ADDRESS OF IDB INTO CRB
          20 AA   00000000'9F   9E   05C2  1029          MOVAB   @#DR$INITIAL,CRB$L_INTD+VEC$L_INITIAL(R10) ; SET ADDRESS OF CONTROLL
                                05CA  1030
                    0A A6   01   90   05CA  1031          MOVB    #DYN$C_ADP,ADP$B_TYPE(R6)       ; TYPE CODE
                       66   54   D0   05CE  1032          MOVL    R4,ADP$L_CSR(R6)                ; SET ADDRESS OF CONFIGURATION REGISTER
                    0C A6   57   B0   05D1  1033          MOVW    R7,ADP$W_TR(R6)                 ; SET TR NUMBER OF ADAPTER
                    0E A6   02   B0   05D5  1034          MOVW    #ATS_DR,ADP$W_ADPTYPE(R6)       ; SET ADAPTER TYPE TO DRA
                    10 A6   5A   D0   05D9  1035          MOVL    R10,ADP$L_CRB(R6)               ; POINT ADP TO CRB
                    28 AA   56   D0   05DD  1036          MOVL    R6,CRB$L_INTD+VEC$L_ADP(R10)    ; SET CRB POINTER TO ADP
                    10 A9   56   D0   05E1  1037          MOVL    R6,IDB$L_ADP(R9)                ; AND INTO IDB
                       55   59   D0   05E5  1038          MOVL    R9,R5                           ; ADDRESS OF IDB
                          040B   30   05E8  1039          BSBW    ADPLINK                         ; LINK ADP TO END OF CHAIN
                00000000'EF      16   05EB  1040          JSB     DR$INITIAL                      ; INITIALIZE MASSBUS ADAPTER
                                05F1  1041          ENBINT                                  ; Re-enable interrupts
                       50   00'  D0   05F4  1042          MOVL    S^#SS$_NORMAL,R0                ; SUCCESSFUL
                                05F7  1043
                    07BC 8F      BA   05F7  1044 999$:   POPR    #^M<R2,R3,R4,R5,R7,R8,R9,R10>; RESTORE ALL REGISTERS
                                05   05FB  1045          RSB                                     ; RETURN
                                05FC  1046
                                05FC  1047 DRACRB:                                         ; SKELETON DRA CRB
                          3C    BB   05FC  1048          PUSHR   #^M<R2,R3,R4,R5>                ; SAVE REGISTERS
                00000000'9F      16   05FE  1049          JSB     @#DR$INT                        ; CALL INTERRUPT DISPATCHER
                          00000000   0604  1050          .LONG   0                               ; CRB$L_INTD+VEC$L_INTD
                          00000000'  0608  1051          .LONG   DR$INITIAL                      ; CRB$L_INTD+VEC$L_INITIAL
                          00000010   060C  1052 DRACRBLEN=.-DRACRB                           ;
```

```
                                060C  1054          .SBTTL  MAXIMIZE ACCESS MODE
                                060C  1055 ;+
                                060C  1056 ; EXE$MAXACMODE - MAXIMIZE ACCESS MODE
                                060C  1057 ;
                                060C  1058 ; THIS ROUTINE IS CALLED TO MAXIMIZE A SPECIFIED ACCESS MODE WITH THE PREVIOUS
                                060C  1059 ; MODE FIELD OF THE CURRENT PSL.
                                060C  1060 ;
                                060C  1061 ; INPUTS:
                                060C  1062 ;
                                060C  1063 ;       R0 = ACCESS MODE TO MAXIMIZE WITH PREVIOUS MODE FIELD OF PSL.
                                060C  1064 ;
                                060C  1065 ; OUTPUTS:
                                060C  1066 ;
                                060C  1067 ;       THE SPECIFIED ACCESS MODE IS MAXIMIZED WITH THE PREVIOUS MODE FIELD
                                060C  1068 ;       OF THE CURRENT PSL AND RETURNED IN REGISTER R0.
                                060C  1069 ;
                                060C  1070 ;       REGISTERS R2 AND R3 ARE PRESERVED ACROSS CALL.
                                060C  1071 ;-
                                060C  1072
                                060C  1073 EXE$MAXACMODE::                          ;MAXIMIZE ACCESS MODE
                          51    060C  1074          MOVPSL  R1                     ;READ CURRENT PSL
     50   51   02   16   ED    060E  1075          CMPZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R0 ;COMPARE WITH PREVIOUS MODE
                          05   0613  1076          BLEQ    10$                    ;IF LEQ SPECIFIED ACCESS MODE LESS PRIVILEGE
     50   51   02   16   EF    0615  1077          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R0 ;EXTRACT PREVIOUS MODE FIELD
                          05   061A  1078 10$:      RSB                            ;
```

```
                     061B  1080          .SBTTL  BUG$CHECK       Bugcheck notification routine
                     061B  1081 ;++
                     061B  1082 ;
                     061B  1083 ; Functional Description:
                     061B  1084 ;
                     061B  1085 ;       This routine types the text of the VMS bugcheck and is equivalent
                     061B  1086 ;       to a supervisor ERRSUP
                     061B  1087 ;
                     061B  1088 ; Calling Sequence:
                     061B  1089 ;       BSBW    BUG$CHECK
                     061B  1090 ;       .ASCIC  "error text"
                     061B  1091 ;
                     061B  1092 ; Input Parameters:
                     061B  1093 ;       @(SP)   -> ASCIC explaination
                     061B  1094 ;
                     061B  1095 ; Implicit Inputs:
                     061B  1096 ;       NONE
                     061B  1097 ;
                     061B  1098 ; Output Parameters:
                     061B  1099 ;       NONE
                     061B  1100 ;
                     061B  1101 ; Implicit Outputs:
                     061B  1102 ;       NONE
                     061B  1103 ;
                     061B  1104 ; Side Effects:
                     061B  1105 ;       NONE
                     061B  1106 ;
                     0613  1107 ;--
                     061B  1108
                     061B  1109 BUG$CHECK::
          50   6E    D0  061B  1110          MOVL    (SP),R0                         ; GET ADDRESS OF TEXT
0000FE00  9F   01    C8  061E  1111          BISL    #DSA$M_HALT,@#DSA$GL_FLAGS ; Force Halt on error        [12]
                         0625  1112          ERRSUP_S ,(R0)
               E3    11  0636  1113          BRB     BUG$CHECK                       ; CAN'T CONTINUE
```

```
0638  1115          .SBTTL  QIO$LOAD_DB     Insert unit into data base
0638  1116 ;++
0638  1117 ; FUNCTIONAL DESCRIPTION:
0638  1118 ;
0638  1119 ;       This routine will load a single unit into an already existing
0638  1120 ;       database.
0638  1121 ;
0638  1122 ; CALLING SEQUENCE:
0638  1123 ;
0638  1124 ;       CALL QIO$LOAD_DB(ACF_LIST)
0638  1125 ;
0638  1126 ; INPUT PARAMETERS:
0638  1127 ;
0638  1128 ;       ACF_LIST -
0638  1129 ;
0638  1130 ;       ACF$L_ADAPTER(AP) = ADDRESS OF ADAPTER CONTROL BLOCK
0638  1131 ;       ACF$L_CONFIGREG(AP) = ADDRESS OF CONFIGURATION STATUS REGISTER
0638  1132 ;       ACF$W_AVECTOR(AP) = OFFSET TO ADAPTER INTERRUPT VECTOR (SCB)
0638  1133 ;       ACF$B_AUNIT(AP) = ADAPTER UNIT NUMBER
0638  1134 ;       ACF$B_AFLAG(AP) = ADAPTER GENERATION CONTROL FLAGS
0638  1135 ;       ACF$L_CONTRLREG(AP) = ADDRESS OF CONTROL REGISTER
0638  1136 ;       ACF$W_CVECTOR(AP) = OFFSET TO CONTROLLER INTERRUPT VECTOR (TABLE)
0638  1137 ;       ACF$B_CUNIT(AP) = CONTROLLER UNIT NUMBER
0638  1138 ;       ACF$B_CNUMVEC(AP) = NUMBER OF CONTROLLER VECTORS
0638  1139 ;       ACF$L_DEVNAME(AP) = ADDRESS OF DEVICE NAME COUNTED STRING
0638  1140 ;       ACF$L_DRVNAME(AP) = ADDRESS OF DRIVER NAME COUNTED STRING
0638  1141 ;
0638  1142 ; IMPLICIT INPUTS:
0638  1143 ;
0638  1144 ;       NONE
0638  1145 ;
0638  1146 ; OUTPUT PARAMETERS:
0638  1147 ;
0638  1148 ;       NONE
0638  1149 ;
0638  1150 ; IMPLICIT OUTPUTS:
0638  1151 ;
0638  1152 ;       DATABASE LOADED
0638  1153 ;
0638  1154 ; COMPLETION CODES:
0638  1155 ;
0638  1156 ;       R0 = STATUS OF OPERATION
0638  1157 ;
0638  1158 ; SIDE EFFECTS:
0638  1159 ;
0638  1160 ;       NONE
0638  1161 ;
0638  1162 ;--
```

```
                          0638  1164 QIO$LOAD_DB:
                     0FFC 0638  1165         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                          063A  1166
          00000004'EF   94 063A  1167         CLRB    L^LOAD_FLAGS            ;CLEAR LOADER FLAGS                     [11]
                0307    30 0640  1168         BSBW    IOGEN$LOCK_IODB         ;LOCK THE I/O DATABASE
                          0643  1169 ;
                          0643  1170 ; LOCATE THE DPT FOR THIS DRIVER
                          0643  1171 ;
           53   18 AC    D0 0643  1172         MOVL    ACF$L_DRVNAME(AP),R3    ; Address of driver name
              52   83    9A 0647  1173         MOVZBL  (R3)+,R2                ; Get length of driver
                 FA21    30 064A  1174         BSBW    LOC$DPT                 ; Locate driver dispatch table
                21 50    E8 064D  1175         BLBS    R0,15$                  ; Branch if found
                          0650  1176
           51   14 AC    9E 0650  1177         MOVAB   ACF$L_DEVNAME(AP),R1    ; ADDRESS OF ASCIC DEVICE NAME
              50   81    9A 0654  1178         MOVZBL  (R1)+,R0                ; GET LENGTH
                   03    BB 0657  1179         PUSHR   #^M<R0,R1>              ; STACK LENGTH,ADDRESS
          00000046'EF    9F 0659  1180         PUSHAB  L^T_NODRIVER            ; ADDRESS OF ASCIC EDT TEXT             [11]
                7E   01  9A 065F  1181         movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                            [13]
                7E   22  98 0662  1182         cvtbl   #ds$k_type_qio_nodriver,-(sp)   ; code                          [13]
     00000000'GF    05    FB 0665  1183         CALLS   #5, G^DSX$PRINT         ; DRIVER FOR DEVICE NOT FOUND           [13]
                   50    D4 066C  1184         CLRL    R0                      ; ERROR
                0267     31 066E  1185         BRW     DB_ERROR
                          0671  1186 ;
                          0671  1187 ; CHECK IF DEVICE DATABASE ALREADY LOADED
                          0671  1188 ;
           55   14 AC    D0 0671  1189 15$:    MOVL    ACF$L_DEVNAME(AP),R5    ;GET ADDR OF DEVICE NAME
              54   85    9A 0675  1190         MOVZBL  (R5)+,R4                ;GET SIZE OF DEVICE NAME
        5A    00000000'GF DE 0678  1191         MOVAL   G^IOC$GL_DEVLIST,R10    ;GET ADDR OF DEVICE LISTHEAD
                   6A    D5 067F  1192 20$:    TSTL    DDB$L_LINK(R10)         ;IS THERE A NEXT DDB?
                   31    13 0681  1193         BEQL    CREATE_DDB              ;BR IF NOT - CREATE A NEW DDB
                5A   6A    D0 0683  1194         MOVL    DDB$L_LINK(R10),R10     ;GET ADDR OF NEXT DDB
              51   14 AA    9E 0686  1195         MOVAB   DDB$T_NAME(R10),R1      ;GET ADDR OF DEVICE NAME
              50   81    9A 068A  1196         MOVZBL  (R1)+,R0                ;GET SIZE OF DEVICE NAME
  65  54   00   61   50    2D 068D  1197         CMPC5   R0,(R1),#0,R4,(R5)      ;COMPARE DEVICE NAMES
                   EA    12 0693  1198         BNEQ    20$                     ;BR IF NOT EQUAL
                          0695  1199
              50   04 AA    D0 0695  1200         MOVL    DDB$L_UCB(R10),R0       ;GET ADDR OF FIRST UCB
              59   20 A0    D0 0699  1201         MOVL    UCB$L_CRB(R0),R9        ;GET ADDR OF CRB
              58   1C A9    D0 069D  1202         MOVL    CRB$L_INTD+VEC$L_IDB(R9),R8 ;GET ADDR OF IDB
           48 A0   12 AC    91 06A1  1203 30$:    CMPB    ACF$B_CUNIT(AP),UCB$W_UNIT(R0) ;IS UCB ALREADY LOADED?
                   03    12 06A6  1204         BNEQ    40$                     ;BR IF NOT
                 0226    31 06A8  1205         BRW     DB_EXIT                 ;ELSE NOTHING TO DO - EXIT
              50   2C A0    D0 06AB  1206 40$:    MOVL    UCB$L_LINK(R0),R0       ;GET ADDR OF NEXT UCB
                   F0    12 06AF  1207         BNEQ    30$                     ;BR IF THERE IS ONE
                 0126    31 06B1  1208         BRW     CREATE_UCB              ;CREATE THE NEW UCB
                          06B4  1209 ;
                          06B4  1210 ; CREATE THE DDB
                          06B4  1211 ;
                          06B4  1212 CREATE_DDB:
                51   34   9A 06B4  1213         MOVZBL  #DDB$K_LENGTH,R1        ;GET SIZE OF DDB
                  0303    30 06B7  1214         BSBW    QIO$ALLOCATE            ;CREATE THE DDB
                03 50     E8 06BA  1215         BLBS    R0,10$                  ;BR IF SUCCESS
                 0218     31 06BD  1216         BRW     LOA_ERROR               ;ELSE - ERROR
     00000004'EF   01    88 06C0  1217 10$:    BISB    #LOAD_M_DDB,L^LOAD_FLAGS ;SET DDB LOADED FLAG                  [11]
     00000005'EF   6A    DE 06C7  1218         MOVAL   DDB$L_LINK(R10),L^DDB_BLINK ;SAVE DDB BACKWARD LINK             [11]
                6A   52    D0 06CE  1219         MOVL    R2,DDB$L_LINK(R10)      ;SET LINK TO THIS DDB
                5A   52    D0 06D1  1220         MOVL    R2,R10                  ;GET ADDR OF DDB
```

```
                08 AA   51    B0  06D4  1221          MOVW    R1,DDB$W_SIZE(R10)       ;SET SIZE
                0A AA   06    90  06D8  1222          MOVB    #DYN$C_DDB,DDB$B_TYPE(R10) ;SET TYPE
                     0B AB    96  06DC  1223          INCB    DPT$B_REFC(R11)          ;INCREMENT DPT REFERENCE COUNT
                50   14 BC    9A  06DF  1224          MOVZBL  @ACF$L_DEVNAME(AP),R0    ;GET SIZE OF DEVICE NAME
                          50    D6  06E3  1225          INCL    R0                       ;ADD 1 BYTE FOR COUNT
          14 AA 14 BC    50    28  06E5  1226          MOVC    R0,@ACF$L_DEVNAME(AP),DDB$T_NAME(R10) ;SET DEVICE NAME
                50   18 BC    9A  06EB  1227          MOVZBL  @ACF$L_DRVNAME(AP),R0    ;GET SIZE OF DRIVER NAME
                          50    D6  06EF  1228          INCL    R0                       ;ADD 1 BYTE FOR COUNT
          24 AA 18 BC    50    28  06F1  1229          MOVC    R0,@ACF$L_DRVNAME(AP),DDB$T_DRVNAME(R10) ;SE  DRIVER NAME
                                   06F7  1230 ;
                                   06F7  1231 ; CHECK IF CRB/IDB NEED TO BE CREATED
                                   06F7  1232 ;
             14 0D AB    00    E0  06F7  1233          BBS     #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),CREATE_CRB ;BR IF SUBCONTROLLER
                0C AB    01    91  06FC  1234          CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
                              0E   13  0700  1235          BEQL    CREATE_CRB               ;BR IF YES
                     50    6C    D0  0702  1236          MOVL    ACF$L_ADAPTER(AP),R0     ;GET ADDR OF ADP
                59   10 A0    D0  0705  1237          MOVL    ADP$L_CRB(R0),R9         ;GET ADDR OF EXISTING CRB
                58   1C A9    D0  0709  1238          MOVL    CRB$L_INTD+VEC$L_IDB(R9),R8 ;GET ADDR OF IDB
                        00CA    31  070D  1239          BRW     CREATE_UCB               ;CREATE A NEW UCB
                                   0710  1240 ;
                                   0710  1241 ; CREATE CRB
                                   0710  1242 ;
                                   0710  1243 CREATE_CRB:
                51   13 AC    9A  0710  1244          MOVZBL  ACF$B_CNUMVEC(AP),R1     ;GET NUMBER OF INT VECTORS
                          51    D7  0714  1245          DECL    R1                       ;ONE IS ALWAYS ASSUMED
                     51   24    A4  0716  1246          MULW    #VEC$K_LENGTH,R1         ;COMPUTE SIZE OF EXTRA DISPATCHERS
                     51   38    A0  0719  1247          ADDW    #CRB$K_LENGTH,R1         ;COMPUTE TOTAL SIZE OF CRB
                        029E    30  071C  1248          BSBW    QIO$ALLOCATE             ;ALLOCATE THE CRB
                     03 50    E8  071F  1249          BLBS    R0,10$                   ;BR IF SUCCESS
                        01B3    31  0722  1250          BRW     DB_ERROR                 ;ELSE - ERROR
00000004'EF          02    88  0725  1251 10$:     BISB    #LOAD_M_CRB,L^LOAD_FLAGS ;SET CRB LOADED FLAG                    [11]
                59   52    D0  072C  1252          MOVL    R2,R9                    ;GET ADDR OF CRB
                C8 A9    51    B0  072F  1253          MOVW    R1,CRB$W_SIZE(R9)        ;SET SIZE
                0A A9    05    90  0733  1254          MOVB    #DYN$C_CRB,CRB$B_TYPE(R9) ;SET TYPE
                69   69    DE  0737  1255          MOVAL   CRB$L_WQFL(R9),CRB$L_WQFL(R9) ;SET WAIT QUEUE LISTHEAD
                04 A9    69    DE  073A  1256          MOVAL   CRB$L_WQFL(R9),CRB$L_WQBL(R9) ;..
             19 0D AB    00    E1  073E  1257          BBC     #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),CREATE_IDB ;BR IF NOT SUBCONTROLLER
                50   6C    D0  0743  1258          MOVL    ACF$L_ADAPTER(AP),R0     ;GET ADDR OF ADAPTER CONTROL BLOCK
                51   10 A0    D0  0746  1259          MOVL    ADP$L_CRB(R0),R1         ;GET ADDR OF SECONDARY CRB
                10 A9    51    D0  074A  1260          MOVL    R1,CRB$L_LINK(R9)        ;SET ADDR OF SECONDARY CRB
                52   1C A1    D0  074E  1261          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R2 ;GET ADDR OF SECONDARY IDB
                53   0A AC    9A  0752  1262          MOVZBL  ACF$B_AUNIT(AP),R3       ;GET ADAPTER UNIT NUMBER
             14 A243 15 A9    9E  0756  1263          MOVAB   CRB$L_INTD+1(R9),IDB$L_UCBLST(R2)[R3] ;SET ADDR OF DISPATCHER
                                   075C  1264 ;
                                   075C  1265 ; CREATE IDB
                                   075C  1266 ;
                                   075C  1267 CREATE_IDB:
                     51   34    9A  075C  1268          MOVZBL  #IDB$K_LENGTH,R1         ;SET LENGTH OF IDB
                        025B    30  075F  1269          BSBW    QIO$ALLOCATE             ;ALLOCATE THE IDB
                     03 50    E8  0762  1270          BLBS    R0,10$                   ;BR IF SUCCESS
                        0170    31  0765  1271          BRW     DB_ERROR                 ;ELSE - ERROR
00000004'EF          04    88  0768  1272 10$:     BISB    #LOAD_M_IDB,L^LOAD_FLAGS ;SET IDB LOADED FLAG                    [11]
                58   52    D0  076F  1273          MOVL    R2,R8                    ;GET ADDR OF IDB
                08 A8    51    B0  0772  1274          MOVW    R1,IDB$W_SIZE(R8)        ;SET SIZE
                0A A8    09    90  0776  1275          MOVB    #DYN$C_IDB,IDB$B_TYPE(R8) ;SET TYPE
                10 A8    6C    D0  077A  1276          MOVL    ACF$L_ADAPTER(AP),IDB$L_ADP(R8) ;SET ADDR OF ADP
                68   0C AC    D0  077E  1277          MOVL    ACF$L_CONTRLREG(AP),IDB$L_CSR(R8) ;SET ADDR OF CSR
```

```
                0C A8   08   9B  0782  1278          MOVZBW  #8,IDB$W_UNITS(R8)      ; Always set up for 8 units
                                0786  1279 ;
                                0786  1280 ; CREATE INTERRUPT VECTOR DISPATCHER(S)
                                0786  1281 ;
                                0786  1282 CREATE_VEC:
                56   10 AC   32   0786  1283          CVTWL   ACF$W_CVECTOR(AP),R6    ;GET VECTOR TABLE OFFSET
                        0A   14   078A  1284          BGTR    5$                      ;BR IF VECTOR SPECIFIED
           50   007C802A 8F   D0   078C  1285          MOVL    #SYSG$_INVVEC,R0        ;SET INVALID VECTOR ERROR
                      0142   31   0793  1286          BRW     DB_ERROR                ;...EXIT
                55   13 AC   9A   0796  1287 5$:      MOVZBL  ACF$B_CNUMVEC(AP),R5    ;GET NUMBER OF INTERRUPT VECTORS
                54   14 A9   DE   079A  1288          MOVAL   CRB$L_INTD(R9),R4       ;GET ADDR OF FIRST DISPATCHER
           64   0000003C'EF   D0   079E  1289 10$:     MOVL    L^MBINT_DISP,VEC$Q_DISPATCH(R4) ;SET 'PUSHR #^M<R2,R3,R4,R5>''   [11]
                                07A5  1290                                          ; AND ''JSB @#''
                14 A4   6C   D0   07A5  1291          MOVL    ACF$L_ADAPTER(AP),VEC$L_ADP(R4) ;SET ADDR OF ADP
                08 A4   58   D0   07A9  1292          MOVL    R8,VEC$L_IDB(R4)        ;SET ADDR OF IDB
                0C AB   01   91   07AD  1293          CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
                        1E   12   07B1  1294          BNEQ    30$                     ;BR IF NOT
                   50   6C   D0   07B3  1295          MOVL    ACF$L_ADAPTER(AP),R0    ;GET ADDR OF ADP
            50     56   10 A0   C1   07B6  1296          ADDL3   ADP$L_VECTOR(R0),R6,R0  ;GET ADDR OF VECTOR TABLE ENTRY
           60   00000000'8F   D1   07BB  1297          CMPL    #UBA$ONEXINT,(R0)       ;IS VECTOR IN USE?
                        0A   13   07C2  1298          BEQL    20$                     ;BR IF NOT
           50   007C801A 8F   D0   07C4  1299          MOVL    #SYSG$_VECINUSE,R0      ;SET ERROR STATUS
                      010A   31   07CB  1300          BRW     DB_ERROR                ;...EXIT
                                07CE  1301 20$:
                    F82F'   30   07CE  1302          BSBW    IOGEN$CONN_VEC          ; CONNECT UB DEVICE VECTOR
                56   04   C0   07D1  1303 30$:      ADDL    #4,R6                   ;INCREMENT VECTOR OFFSET
                54   24   C0   07D4  1304          ADDL    #VEC$K_LENGTH,R4        ;INCREMENT DISPATCH ADDR
                  C4 55   F5   07D7  1305          SOBGTR  R5,10$                  ;DECREMENT VECTOR COUNT - BR IF MORE
                                07DA  1306 ;
                                07DA  1307 ; CREATE A UCB
                                07DA  1308 ;
                                07DA  1309 CREATE_UCB:
             51   0E AB   3C   07DA  1310          MOVZWL  DPT$W_UCBSIZE(R11),R1   ;GET SIZE OF UCB
                    01DC   30   07DE  1311          BSBW    QIO$ALLOCATE            ;ALLOCATE THE UCB
                03 50   E8   07E1  1312          BLBS    R0,5$                   ;BR IF SUCCESS
                   00F1   31   07E4  1313          BRW     DB_ERROR                ;ELSE - ERROR
      00000004'EF   08   88   07E7  1314 5$:      BISB    #LOAD_M_UCB,L^LOAD_FLAGS ;SET UCB LOADED FLAG                    [11]
                57   52   D0   07EE  1315          MOVL    R2,R7                   ;GET ADDR OF UCB
             08 A7   51   B0   07F1  1316          MOVW    R1,UCB$W_SIZE(R7)       ;SET SIZE
             0A A7   10   90   07F5  1317          MOVB    #DYN$C_UCB,UCB$B_TYPE(R7) ;SET TYPE
          0C A7   0C A7   DE   07F9  1318          MOVAL   UCB$L_ASTQFL(R7),UCB$L_ASTQFL(R7) ;SET AST QUEUE LISTHEAD
          10 A7   0C A7   DE   07FE  1319          MOVAL   UCB$L_ASTQFL(R7),UCB$L_ASTQBL(R7) ;...
             20 A7   59   D0   0803  1320          MOVL    R9,UCB$L_CRB(R7)        ;SET ADDR OF CRB
                   0C A9   B6   0807  1321          INCW    CRB$W_REFC(R9)          ;INCREMENT CRB REFERENCE COUNT
                50   12 AC   9A   080A  1322          MOVZBL  ACF$B_CUNIT(AP),R0      ;GET CONTROLLER UNIT NUMBER
              14 A840    57   D0   080E  1323          MOVL    R7,IDB$L_UCBLST(R8)[R0] ;SET ADDR OF UCB IN IDB
             24 A7    5A   D0   0813  1324          MOVL    R10,UCB$L_DDB(R7)       ;SET ADDR OF DDB
          40 A7   40 A7   DE   0817  1325          MOVAL   UCB$L_IOQFL(R7),UCB$L_IOQFL(R7) ;SET I/O QUEUE LISTHEAD
          44 A7   40 A7   DE   081C  1326          MOVAL   UCB$L_IOQFL(R7),UCB$L_IOQBL(R7) ;...
          48 A7   12 AC   9B   0821  1327          MOVZBW  ACF$B_CUNIT(AP),UCB$W_UNIT(R7) ;SET UNIT NUMBER
          74 A7   0A AC   90   0826  1328          MOVB    ACF$B_AUNIT(AP),UCB$B_SLAVE(R7) ;SET SLAVE CNTRLER NUMBER
                                082B  1329
             51   04 AA   DE   082B  1330          MOVAL   DDB$L_UCB(R10),R1       ;GET ADDR OF ADDR OF FIRST UCB
                50   61   D0   082F  1331          MOVL    (R1),R0                 ;GET ADDR OF FIRST UCB
                        09   13   0832  1332          BEQL    20$                     ;BR IF NONE
             51   2C A0   DE   0834  1333 10$:      MOVAL   UCB$L_LINK(R0),R1       ;GET ADDR OF ADDR OF NEXT UCB
                50   61   D0   0838  1334          MOVL    (R1),R0                 ;GET ADDR OF NEXT UCB
```

```
                F7   12   083B  1335          BNEQ    10$                     ;BR IF IT EXISTS
    00000009'EF   51   D0   083D  1336 20$:    MOVL    R1,L^UCB_BLINK          ;SAVE UCB BACKWARD LINK                 [11]
          61   57   D0   0844  1337          MOVL    R7,(R1)                 ;SET FORWARD LINK TO UCB
                            0847  1338
                            0847  1339 ;
                            0847  1340 ; INITIALIZE ALL THE CONTROL BLOCKS
                            0847  1341 ;
                            0847  1342 INIT_DB:
          54   5B   D0   0847  1343          MOVL    R11,R4                  ;SET ADDR OF DRIVER PROLOGUE
          55   57   D0   084A  1344          MOVL    R7,R5                   ;SET ADDR OF UCB
    00000000'EF   16   084D  1345          JSB     IOC$.NITDRV                     ; Init the control blocks       [11]
 58 A5    0800 8F   A8   0853  1346          BISW    #UCB$M_VALID,UCB$W_STS(R5);FORCE SOFTWARE VOLUME VALID
          0A 50   E8   0859  1347          BLBS    R0,INIT_CNTRL           ;BR IF SUCCESS
 50   007C800A 8F   D0   085C  1348          MOVL    #SYSG$_INVDPTINI,R0     ;SET ERROR STATUS
            0072   31   0863  1349          BRW     DB_ERROR                ;...EXIT
                            0866  1350 ;
                            0866  1351 ; CALL THE CONTROLLER AND DRIVE INITIALIZATION ROUTINES
                            0866  1352 ;
                            0866  1353 INIT_CNTRL:
                            0866  1354          DSBINT                          ;DISABLE INTERRUPTS
06 00000004'EF   01   E1   086C  1355          BBC     #LOAD_V_CRB,L^LOAD_FLAGS,INIT_UNIT ;BR IF CRB NOT JUST CREATED  [11]
          56   5A   D0   0874  1356          MOVL    R10,R6                  ;SET ADDR OF DDB
            00D8   30   0877  1357          BSBW    IOGEN$CNTRL_INI         ;INITIALIZE THE CONTROLLER
                            087A  1358
                            087A  1359 INIT_UNIT:
          55   57   D0   087A  1360          MOVL    R7,R5                   ;GET ADDR OF UCB
          54   68   D0   087D  1361          MOVL    IDB$L_CSR(R8),R4        ;GET ADDR OF CSR
          53   54   D0   0880  1362          MOVL    R4,R3                   ;ASSUME ONLY ONE CSR
    50    10 A9   D0   0883  1363          MOVL    CRB$L_LINK(R9),R0       ;GET ADDR OF SECONDARY CRB
          07   13   0887  1364          BEQL    10$                     ;BR IF NONE
    50    1C A0   D0   0889  1365          MOVL    CRB$L_INTD+VEC$L_IDB(R0),R0 ;GET ADDR OF SECONDARY IDB
          54   60   D0   088D  1366          MOVL    IDB$L_CSR(R0),R4        ;GET SECONDARY CSR ADDR
 0C AB    01   91   0890  1367 10$:    CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
          15   12   0894  1368          BNEQ    20$                     ;BR IF NOT
          56   6C   D0   0896  1369          MOVL    ACF$L_ADAPTER(AP),R6    ;GET ADDR OF ADP
          56   66   D0   0899  1370          MOVL    ADP$L_CSR(R6),R6        ;GET ADDR OF adapter CSR
          50   56   D0   089C  1371          MOVL    R6,R0
             F75E'   30   089F  1372          BSBW    IO$TESTCSR_L            ; CHECK FOR NONEX CSR
          06 50   E8   08A2  1373          BLBS    R0,20$                  ; BRANCH IF CSR EXISTS
 58 A7    10   AA   08A5  1374          BICW    #UCB$M_ONLINE,UCB$W_STS(R7) ;SET DEVICE OFFLINE
          23   11   08A9  1375          BRB     30$                     ;
 50    0C AA   D0   08AB  1376 20$:    MOVL    DDB$L_DDT(R10),R0       ; GET ADDRESS OF DDT
 51    18 A0   D0   08AF  1377          MOVL    DDT$L_UNITINIT(R0),R1   ; GET OFFSET OF UNIT CODE
 03 51    10   E0   08B3  1378          BBS     #16,R1,25$              ; BRANCH IF ABSOLUTE SUPERVISOR ADDR
    51    50   C0   08B7  1379          ADDL    R0,R1                   ; ADD IN BASE ADDRESS
00000000'8F   51   D1   08BA  1380 25$:    CMPL    R1,#IOC$RTN             ; IF EQUAL TO IOC$RETURN THEN
          06   12   08C1  1381          BNEQ    27$                     ; UNIT INIT WASN'T SPECIFIED
 51    2C A9   D0   08C3  1382          MOVL    CRB$L_INTD+VEC$L_UNITINIT(R9),R1  ; SO GET IT FROM CRB
          05   13   08C7  1383          BEQL    30$                     ; NOT SPECIFIED IN CRB EITHER
          61   16   08C9  1384 27$:    JSB     (R1)                    ; CALL UNIT INITIALIZATION
          56   5A   D0   08CB  1385          MOVL    R10,R6                  ; SET UP ADDRESS OF DDB
                            08CE  1386 30$:
                            08CE  1387          ENBINT                          ;RE-ENABLE INTERRUPTS
                            08D1  1388 ;
                            08D1  1389 ; OPERATION COMPLETED - UNLOCK I/O DATABASE AND EXIT
                            08D1  1390 ;
                            08D1  1391 DB_EXIT:
```

```
                 007A   30  08D1  1392          BSBW    IOGEN$UNLK_IODB          ;UNLOCK I/O DATABASE
                                08D4  1393                                       ; AND LOWER IPL
           50     01   D0  08D4  1394          MOVL    #1,R0                    ;SET SUCCESS
                       04  08D7  1395          RET                              ;...EXIT
```

```
                                08D8  1397          .SBTTL  DB_ERROR        ERROR LOADING DATABASE
                                08D8  1398 ;++
                                08D8  1399 ;
                                08D8  1400 ; DB_ERROR - LOCAL ROUTINE TO UNLOAD A PARTIALLY LOADED DATABASE
                                08D8  1401 ;
                                08D8  1402 ;       This routine checks the loading flags (LOAD_FLAGS) and
                                08D8  1403 ;       unload and unlinks any control blocks that were just created.
                                08D8  1404 ;
                                08D8  1405 ; INPUTS:
                                08D8  1406 ;
                                08D8  1407 ;       LOAD_FLAGS = FLAGS INDICATING CONTROL BLOCKS THAT HAVE JUST
                                08D8  1408 ;                    BEEN CREATED
                                08D8  1409 ;
                                08D8  1410 ; OUTPUTS:
                                08D8  1411 ;
                                08D8  1412 ;       R0 = ERROR STATUS THAT CAUSED UNLOADING
                                08D8  1413 ;
                                08D8  1414 ;--
                                08D8  1415 DB_ERROR:
                     50   DD    08D8  1416          PUSHL   R0                      ;SAVE ERROR STATUS
                                08DA  1417 ;
                                08DA  1418 ; UCB UNLOADING
                                08DA  1419 ;
          1C 00000004'EF   03   E1   08DA  1420          BBC     #LOAD_V_UCB,L^LOAD_FLAGS,20$ ;BR IF UCB NOT CREATED     [11]
                00000009'FF   D4   08E2  1421          CLRL    @L^UCB_BLINK            ;CLEAR POINTER TO UCB           [11]
                     0C A9    B7   08E8  1422          DECW    CRB$W_REFC(R9)          ;DECREMENT CRB REFERENCE COUNT
                   50   12 AC   9A   08EB  1423          MOVZBL  ACF$B_CUNIT(AP),R0      ;GET CONTROLLER UNIT NUMBER
                      14 A840   D4   08EF  1424          CLRL    IDB$L_UCBLST(R8)[R0]    ;CLEAR IDB POINTER TO UCB
                      50   57   D0   08F3  1425          MOVL    R7,R0                   ;SET ADDR OF UCB
                           02   10   08F6  1426          BSBB    10$                     ;DEALLOCATE THE UCB
                           04   11   08F8  1427          BRB     20$                     ;
                                08FA  1428 ;
                                08FA  1429 ; LOCAL SUBROUTINE TO DEALLOCATE A CONTROL BLOCK
                                08FA  1430 ;
                                08FA  1431 ;       INPUT - R0 = ADDRESS OF BLOCK TO DEALLOCATE
                                08FA  1432 ;
                     00E3   30   08FA  1433 10$:    BSBW    QIO$DEALLOCATE          ;DEALLOCATE TO NON-PAGED POOL
                           05   08FD  1434          RSB
                                08FE  1435
                                08FE  1436 ;
                                08FE  1437 ; IDB UNLOADING
                                08FE  1438 ;
          05 00000004'EF   02   E1   08FE  1439 20$:    BBC     #LOAD_V_IDB,L^LOAD_FLAGS,30$ ;BR IF IDB NOT CREATED     [11]
                      50   58   D0   0906  1440          MOVL    R8,R0                   ;SET ADDR OF IDB
                           EF   10   0909  1441          BSBB    10$                     ;DEALLOCATE THE IDB
                                090B  1442 ;
                                090B  1443 ; CRB UNLOADING
                                090B  1444 ;
          1A 00000004'EF   01   E1   090B  1445 30$:    BBC     #LOAD_V_CRB,L^LOAD_FLAGS,50$ ;BR IF CRB NOT CREATED     [11]
                10 0D AB   00   E1   0913  1446          BBC     #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),40$ ;BR IF NOT SUBCONTROLLER
                      51   10 A9   D0   0918  1447          MOVL    CRB$L_LINK(R9),R1       ;GET ADDR OF SECONDARY CRB
                      51   1C A1   D0   091C  1448          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R1 ;GET ADDR OF SECONDARY IDB
                      50   0A AC   9A   0920  1449          MOVZBL  ACF$B_AUNIT(AP),R0      ;GET ADAPTER UNIT NUMBER
                         14 A140   D4   0924  1450          CLRL    IDB$L_UCBLST(R1)[R0]    ;CLEAR DISPATCHER POINTER
                         50   59   D0   0928  1451 40$:    MOVL    R9,R0                   ;SET ADDR OF CRB
                              CD   10   092B  1452          BSBB    10$                     ;DEALLOCATE THE CRB
                                092D  1453 ;
```

```
                          092D  1454 ; DDB UNLOADING
                          092D  1455 ;
      0E 00000004'EF   00   E1  092D  1456 50$:    BBC     #LOAD_V_DDB,L^LOAD_FLAGS,60$ ;BR IF DDB NOT CREATED          [11]
                    0B AB   97  0935  1457         DECB    DPT$B_REFC(R11)              ;DECREMENT DPT REFERENCE COUNT
               00000005'FF   D4  0938  1458         CLRL    @L^DDB_BLINK                 ;CLEAR LINK TO THE DDB           [11]
                      50   5A  D0  093E  1459         MOVL    R10,R0                       ;SET ADDR OF DDB
                           B7  10  0941  1460         BSBB    10$                          ;DEALLOCATE THE DDB
                               0943  1461 60$:
                      0008   30  0943  1462         BSBW    IOGEN$UNLK_IODB              ;UNLOCK THE I/O DATABASE
                               0946  1463                                              ; AND LOWER IPL
                        50 8ED0  0946  1464         POPL    R0                           ;RESTORE STATUS
                           04  0949  1465         RET                                  ;...EXIT
                               094A  1466
                               094A  1467
                               C94A  1468
                               094A  1469         .SBTTL                  I/O DATABASE INTERLOCKS
                               094A  1470 IOGEN$LOCK_IODB:
                               094A  1471         SETIPL  #31
                           05  094D  1472         RSB
                               094E  1473 IOGEN$UNLK_IODB:
                               094E  1474         SETIPL  #0
                           05  0951  1475         RSB
```

```
                                0952  1477          .SBTTL  IOGEN$CNTRL_INI Initialize a controller
                                0952  1478 ;++
                                0952  1479 ;
                                0952  1480 ; IOGEN$CNTRL_INI - INITIALIZE THE DEVICE CONTROLLER
                                0952  1481 ;
                                0952  1482 ; INPUTS - R6 = ADDR OF DDB
                                0952  1483 ;           R11 = ADDR OF DRIVER PROLOGUE TABLE
                                0952  1484 ;
                                0952  1485 ;--
                                0952  1486 IOGEN$CNTRL_INI::
0C AB    01    91              0952  1487          CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
           3F    12              0956  1488          BNEQ    20$                         ;BR IF NOT
   OFFC 8F     BB              0958  1489          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
50   04 A6     D0              095C  1490          MOVL    DDB$L_UCB(R6),R0            ;GET ADDR OF FIRST UCB
58   20 A0     D0              0960  1491          MOVL    UCB$L_CRB(R0),R8            ;GET ADDR OF CRB
55   1C A8     D0              0964  1492          MOVL    CRB$L_INTD+VEC$L_IDB(R8),R5 ;SET ADDR OF IDB
   54   65     D0              0968  1493          MOVL    IDB$L_CSR(R5),R4            ;SET ADDR OF CSR
50   10 A5     D0              096B  1494          MOVL    IDB$L_ADP(R5),R0            ;GET ADDR OF ADP
   50   60     D0              096F  1495          MOVL    ADP$L_CSR(R0),R0            ;GET ADDR OF UBA CSR
   0040 8F     BB              0972  1496          PUSHR   #^M<R6>                     ;SAVE DDB ADDRESS
56   10 A5     D0              0976  1497          MOVL    IDB$L_ADP(R5),R6            ;GET ADDRESS OF ADP
   56   66     D0              097A  1498          MOVL    ADP$L_CSR(R6),R6            ;GET ADDRESS OF CSR
   50   54     D0              097D  1499          MOVL    R4,R0                       ;Copy address of Controller CSR
      F67D'     30              0980  1500          BSBW    IO$TESTCSR_W                ;TEST IF CONTROLLER EXISTS
   0040 8F     BA              0983  1501          POPR    #^M<R6>                     ;RESTORE DDB ADDRESS
     08 50     E9              0987  1502          BLBC    R0,10$                      ;BRANCH IF NON-EXISTENT CSR
50   20 A8     D0              098A  1503          MOVL    CRB$L_INTD+VEC$L_INITIAL(R8),R0 ;GET ADDR OF INIT ROUTINE
           02    13              098E  1504          BEQL    10$                         ;BR IF NONE
                                0990  1505                                              ;R4 = ADDR OF CSR
                                0990  1506                                              ;R5 = ADDR OF IDB
                                0990  1507                                              ;R8 = ADDR OF CRB
           60    16              0990  1508          JSB     (R0)                        ;INIT CONTROLLER
                                0992  1509 10$:
   OFFC 8F     BA              0992  1510          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;RESTORE REGISTERS
                05              0996  1511          RSB
                                0997  1512 20$:
0C AB    00    91              0997  1513          CMPB    #AT$_MBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
           1F    12              099B  1514          BNEQ    40$                         ; Branch if not MBA
   OFFC 8F     BB              099D  1515          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
50   04 A6     D0              09A1  1516          MOVL    DDB$L_UCB(R6),R0            ;GET ADDR OF FIRST UCB
58   20 A0     D0              09A5  1517          MOVL    UCB$L_CRB(R0),R8            ;GET ADDR OF CRB
55   1C A8     D0              09A9  1518          MOVL    CRB$L_INTD+VEC$L_IDB(R8),R5 ;SET ADDR OF IDB
   54   65     D0              09AD  1519          MOVL    IDB$L_CSR(R5),R4            ;SET ADDR OF CSR
50   20 A8     D0              09B0  1520          MOVL    CRB$L_INTD+VEC$L_INITIAL(R8),R0 ;GET ADDR OF INIT ROUTINE
           02    13              09B4  1521          BEQL    30$                         ;BR IF NONE
                                09B6  1522                                              ;R4 = ADDR OF CSR
                                09B6  1523                                              ;R5 = ADDR OF IDB
                                09B6  1524                                              ;R8 = ADDR OF CRB
           60    16              09B6  1525          JSB     (R0)                        ;INIT CONTROLLER
                                09B8  1526 30$:
   OFFC 8F     BA              09B8  1527          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;RESTORE REGISTERS
                                09BC  1528 40$:
                05              09BC  1529          RSB
```

```
                        09BD  1531          .SBTTL                  MEMORY ALLOCATION SUBROUTINES
                        09BD  1532 ;+
                        09BD  1533 ; ALONONPAGED CALLS EXE$ALONONPAGED TO ALLOCATE NON-PAGED POOL.
                        09BD  1534 ; ZERO'S THE BLOCK WHEN ALLOCATED
                        09BD  1535 ; ALSO THE LENGTH OF THE ALLOCATED BLOCK IS ADDED TO QIO$L_ALLOCATED.
                        09BD  1536 ;
                        09BD  1537 ; INPUT:
                        09BD  1538 ;       R1 - SIZE OF REQUESTED BLOCK IN BYTES
                        09BD  1539 ;
                        09BD  1540 ; OUTPUTS:
                        09BD  1541 ;       R0 - SUCCESS/FAILURE INDICATION
                        09BD  1542 ;       R1 - ALLOCATED SIZE OF BLOCK
                        09BD  1543 ;       R2 - ADDRESS OF BLOCK
                        09BD  1544 ;-
                        C9BD  1545 QIO$ALLOCATE:
              08   BB   09BD  1546          PUSHR   #^M<R3>                 ; SAVE R3
00000000'EF    16   09BF  1547          JSB     EXE$ALONONPAGED         ; ATTEMPT TO ALLOCATE
              08   BA   09C5  1548          POPR    #^M<R3>                 ; RESTORE R3
           15 50   E9   09C7  1549          BLBC    R0,20$                  ; RETURN W/ ERROR
              06   BB   09CA  1550          PUSHR   #^M<R1,R2>              ; SAVE REGS
51   51   FD 8F   78   09CC  1551          ASHL    #-3,R1,R1               ; DIVIDE BY 8
              82   7C   09D1  1552 10$:     CLRQ    (R2)+                   ; CLEAR
           FB 51   F5   09D3  1553          SOBGTR  R1,10$                  ; COUNT QUADWORD AND CONTINUE
              06   BA   09D6  1554          POPR    #^M<R1,R2>              ; RESTORE
00000000'EF    51   C0   09D8  1555          ADDL    R1,L^QIO$L_ALLOCATE     ; REMEMBER IT WAS ALLOCATE              [11]
              05   09DF  1556 20$:     RSB                             ; RETURN
                   09E0  1557
                   09E0  1558
                   09E0  1559
                   09E0  1560 ;+
                   09E0  1561 ; QIO$DEALLOCATE CALLS EXE$DEANONPAGED TO RELEASE THE MEMORY
                   09E0  1562 ; THE SIZE OF THE BLOCK IS SUBTRACTED FROM QIO$L_ALLOCATE
                   09E0  1563 ; INPUT:        R0 Address of block
                   09E0  1564 ;-
                   09E0  1565 QIO$DEALLOCATE::
           0E   BB   09E0  1566          PUSHR   #^M<R1,R2,R3>           ; SAVE REGS
51   08 A0   3C   09E2  1567          MOVZWL  8(R0),R1                ; GET LENGH OF BLOCK
00000000'EF    51   C2   09E6  1568          SUBL    R1,L^QIO$L_ALLOCATE     ; SUBTRACT FROM SPACE ALLOCATE          [11]
00000000'EF    16   09ED  1569          JSB     EXE$DEANONPAGED         ; RELEASE IT
           0E   BA   09F3  1570          POPR    #^M<R1,R2,R3>           ; RESTORE
              05   09F5  1571          RSB                             ; RETURN
```

```
                        09F6  1573          .SBTTL  ADPLINK          Link adapter to end of ADP list
                        09F6  1574 ;+
                        09F6  1575 ; ADPLINK LINKS THE ADAPTER CONTROL BLOCK  TO THE END OF THE ADP LIST
                        09F6  1576 ;
                        09F6  1577 ; INPUT:
                        09F6  1578 ;       R6 - ADDRESS OF NEW ADP
                        09F6  1579 ; OUTPUTS:
                        09F6  1580 ;       ADP IS LINK TO THE END OF THE ADPLIST LOCATED BY IOC$GL_ADPLIST.
                        09F6  1581 ;-
                        09F6  1582 ADPLINK:                                             ;
          50  FFFFFFFC'EF  9E    09F6  1583          MOVAB   <IOC$GL_ADPLIST-ADP$L_LINK>,R0  ; START OF LIST
           51  04 A0     D0    09FD  1584 10$:      MOVL    ADP$L_LINK(R0),R1           ; FLINK TO FIRST ENTRY
                  05      13    0A01  1585          BEQL    20$                         ; AT END
              50  51      D0    0A03  1586          MOVL    R1,R0                       ; TRY AGAIN
                  F5      11    0A06  1587          BRB     10$                         ;
           04 A0  56      D0    0A08  1588 20$:      MOVL    R6,ADP$L_LINK(R0)           ; CHAIN NEW ADP TO END OF LIST
                  05      0A0C  1589          RSB
                          0A0D  1590
                          0A0D  1591
                          0A0D  1592 ;+
                          0A0D  1593 ; FIND_ADP LOCATES THE ADAPTER CONTROL BLOCK WITH THE MATCHING TR NUMBER
                          0A0D  1594 ; INPUT:         R4=CSR                                          [18]
                          0A0D  1595 ; OUTPUT:        R6=ADP
                          0A0D  1596 ;-
                          0A0D  1597 FIND_ADP:
                50        D4    0A0D  1598          CLRL    R0                          ; INITIALLY A FAILURE
          56  FFFFFFFC'EF  9E    0A0F  1599          MOVAB   L^IOC$GL_ADPLIST-ADP$L_LINK,R6  ; HEAD OF ADAPTER LIST   [11]
           56  04 A6     D0    0A16  1600 10$:      MOVL    ADP$L_LINK(R6),R6           ; ADVANCE TO NEXT IN LIST
                  08      13    0A1A  1601          BEQL    20$                         ; BRANCH IF END OF LIST
              54  66      D1    0A1C  1602          CMPL    ADP$L_CSR(R6),R4            ; CSR NUMBER MATCH?     [18]
                  F5      12    0A1F  1603          BNEQ    10$                         ; BRANCH IF NO MATCH
              50  00'     D0    0A21  1604          MOVL    S^#SS$_NORMAL,R0            ; SUCCESS
                          05    0A24  1605 20$:      RSB
```

```
                     0A25  1607          .Sbttl  Exe$PwrTimChk - Check reasonable interval since power recovery ;[14]
                     0A25  1608 ;++                                                                               [14]
                     0A25  1609 ; Functional Description:                                                         [14]
                     0A25  1610 ;       EXE$PWRTIMCHK is called by driver initialization code to check for        [14]
                     0A25  1611 ;       a sufficient interval since power recovery to expect devices to be        [14]
                     0A25  1612 ;       ready again.  If the return from EXE$PWRTIMCHK indicates that the         [14]
                     0A25  1613 ;       reasonable interval has not yet elapsed, the device driver may elect      [14]
                     0A25  1614 ;       to wait for a while using EXE$PWRTIMCHK check the time.                   [14]
                     0A25  1615 ;                                                                                 [14]
                     0A25  1616 ; Calling Sequence:                                                               [14]
                     0A25  1617 ;       BSB/JSB EXE$PWRTIMCHK                                                     [14]
                     0A25  1618 ;                                                                                 [14]
                     0A25  1619 ; Output Parameters:                                                              [14]
                     0A25  1620 ;       R0 - Low bit clear if interval expired.                                   [14]
                     0A25  1621 ;--                                                                               [14]
```

```
                          0A25  1623 Exe$PwrTimChk::                                ;                                                  [14]
                          0A25  1624          CLRL    R0                            ; Assume interval expired                          [14]
                    50    D4    0A25  1624          CLRL    R0                      ; Assume interval expired                          [14]
                    7F    DB    0A27  1625          MFPR    #PR$_TODR,-(SP)         ; Get current time of day                          [14]
                8E  00000004'EF   D1    0A2A  1626          CMPL    L^EXE$GL_PWRDONE,(SP)+  ; Check for time expired                   [14]
                    02    1B    0A31  1627          BLEQU   10$                     ; Exit with low bit clear if expired               [14]
                    50    D6    0A33  1628          INCL    R0                      ; Else set low bit of R0                           [14]
                          05    0A35  1629 10$:     RSB                             ;                                                  [14]
                          0A36  1630          .END
```

```
$$ARGS                  =  00000007     D
$$T1                    =  00000020     D
$ER                     =  00000006     D
$MODULE                    00000000 R   D   02
$_KNOWN                 =  00000006     D
^CF                        FFFFFFC0     D
ACF$B_AFLAG                0000000B     D
ACF$B_AUNIT                0000000A     D
ACF$B_CNUMVEC              00000013     D
ACF$B_CUNIT                00000012     D
ACF$C_LENGTH               00000020     D
ACF$K_LENGTH               00000020     D
ACF$L_ADAPTER              00000000     D
ACF$L_CONFIGREG            00000004     D
ACF$L_CONTRLREG            0000000C     D
ACF$L_DEVNAME              00000014     D
ACF$L_DRVNAME              00000018     D
ACF$W_AVECTOR              00000008     D
ACF$W_CVECTOR              00000010     D
ACF$W_MAXUNITS             0000001C     D
ADP$B_NUMBER               0000000B     D
ADP$B_PORT                 00000020     D
ADP$B_TYPE                 0000000A     D
ADP$C_DRADPLEN             00000014     D
ADP$C_MBAADPLEN            00000014     D
ADP$C_MPMADPLEN            00000070     D
ADP$C_UBAADPLEN            00000070     D
ADP$K_DRADPLEN             00000014     D
ADP$K_MBAADPLEN            00000014     D
ADP$K_MPMADPLEN            00000070     D
ADP$K_UBAADPLEN            00000070     D
ADP$L_CRB                  00000010     D
ADP$L_CSR                  00000000     D
ADP$L_DPQBL                00000018     D
ADP$L_DPQFL                00000014     D
ADP$L_INTD                 00000064     D
ADP$L_LINK                 00000004     D
ADP$L_MRQBL                00000020     D
ADP$L_MRQFL                0000001C     D
ADP$L_PRQQBL               00000018     D
ADP$L_PRQQFL               00000014     D
ADP$L_SHB                  0000001C     D
ADP$L_VECTOR               00000010     D
ADP$W_ADPTYPE              0000000E     D
ADP$W_DPBITMAP             00000024     D
ADP$W_MRBITMAP             00000026     D
ADP$W_SIZE                 00000008     D
ADP$W_TR                   0000000C     D
ADPLINK                    000009F6 R   D   04
ALLOC$_ACMODE           =  00000010     D
ALLOC$_DEVNAM           =  00000004     D
ALLOC$_NARGS            =  00000004     D
ALLOC$_PHYBUF           =  0000000C     D
ALLOC$_PHYLEN           =  00000008     D
AT$_DR                  =  00000002     D
AT$_MBA                 =  00000000     D
AT$_UBA                 =  00000001     D
AUNKFUNC                   00000008 R   D   02
BIT...                  =  00000004     D
BUG$CHECK                  0000061B RG  D   04
CCB$B_AMOD                 00000009     D
CCB$B_STS                  00000008     D
CCB$C_LENGTH               00000010     D
CCB$K_LENGTH               00000010     D
CCB$L_DIRP                 0000000C     D
CCB$L_UCB                  00000000     D
CCB$L_WIND                 00000004     D
CCB$W_IOC                  0000000A     D
CMK$_                   =  00000004     D
CMK$_ASTEXIT            =  00000000     D
CMK$_CANTIM             =  0000000B     D
CMK$_HIBER              =  00000007     D
CMK$_INITSCB            =  00000008     D
CMK$_LAST               =  0000000E     D
CMK$_MMENABLE           =  00000003     D
CMK$_RCONSOLE           =  00000001     D
CMK$_REFRESH            =  00000005     D
CMK$_SCHDWK             =  00000006     D
CMK$_SETIMR             =  0000000C     D
CMK$_SETIPL             =  00000009     D
CMK$_SETPRT             =  0000000D     D
CMK$_SGIPR              =  0000000A     D
CMK$_TCONSOLE           =  00000002     D
CRB$B_MASK                 0000000E     D
CRB$B_TYPE                 0000000A     D
CRB$C_LENGTH               00000038     D
CRB$K_LENGTH               00000038     D
CRB$L_INTD                 00000014     D
CRB$L_INTD2                00000038     D
CRB$L_LINK                 00000010     D
CRB$L_WQBL                 00000004     D
CRB$L_WQFL                 00000000     D
CRB$W_REFC                 0000000C     D
CRB$W_SIZE                 00000008     D
CREATE_CRB                 00000710 R   D   04
CREATE_DDB                 000006B4 R   D   04
CREATE_IDB                 0000075C R   D   04
CREATE_UCB                 000007DA R   D   04
CREATE_VEC                 00000786 R   D   04
CTL$GW_CHINDX              0000003A RG  D   02
CTL$GW_NMIOCH              00000038 RG  D   02
DB_ERROR                   000008D8 R   D   04
DB_EXIT                    000008D1 R   D   04
DDB$B_ACPCLASS             00000013     D
DDB$B_TYPE                 0000000A     D
DDB$C_LENGTH               00000034     D
DDB$K_LENGTH               00000034     D
DDB$L_ACPD                 00000010     D
DDB$L_DDT                  00000C0C     D
DDB$L_LINK                 00000000     D
DDB$L_UCB                  00000004     D
DDB$T_DRVNAME              00000024     D
DDB$T_NAME                 00000014     D
DDB$W_SIZE                 00000008     D
```

QIO
Symbol table

```
DDB_BLINK                       00000005 R   D  03
DDT$L_ALTSTART                  0000001C     D
DDT$L_CANCEL                    0000000C     D
DDT$L_FDT                       00000008     D
DDT$L_REGDUMP                   00000010     D
DDT$L_START                     00000000     D
DDT$L_UNITINIT                  00000018     D
DDT$L_UNSOLINT                  00000004     D
DDT$W_DIAGBUF                   00000014     D
DDT$W_ERRORBUF                  00000016     D
DIR...                        = FFFFFFFF     D
DPT$B_ADPTYPE                   0000000C     D
DPT$B_FLAGS                     0000000D     D
DPT$B_REFC                      0000000B     D
DPT$B_TYPE                      0000000A     D
DPT$C_LENGTH                    0000002A     D
DPT$K_LENGTH                    0000002A     D
DPT$L_BLINK                     00000004     D
DPT$L_FLINK                     00000000     D
DPT$T_NAME                      0000001E     D
DPT$V_SUBCNTRL                = 00000000     D
DPT$W_INITTAB                   00000010     D
DPT$W_MAXUNITS                  00000016     D
DPT$W_REINITTAB                 00000012     D
DPT$W_SIZE                      00000008     D
DPT$W_UCBSIZE                   0000000E     D
DPT$W_UNLOAD                    00000014     D
DPT$W_VERSION                   00000018     D
DR$INITIAL                      ********   X   00
DR$INT                          ********   X   00
DR780$B_BR                      00000033     D
DR780$B_CONFIG                  00000035     D
DR780$B_SELF                    00000034     D
DR780$B_TR                      00000032     D
DR780$B_UUT                     00000036     D
DR780$K_LEN                     00000037     D
DRACRB                          000005FC R   D  04
DRACRBLEN                     = 00000010     D
DS$GPHARD                       ********   X   00
DS$K_CCB                        ********   X   00
DS$K_ERROR                    = 00000002     D
DS$K_NORMAL                   = 00000001     D
DS$K_PRINTB                   = 00000002     D
DS$K_PRINTF                   = 00000001     D
DS$K_PRINTI                   = 00000000     D
DS$K_PRINTX                   = 00000003     D
DS$K_SEVERE                   = 00000004     D
DS$K_SUBSYS                   = 00000066     D
DS$K_TYPE_ABORT_PROGRAM       = 00000014     D
DS$K_TYPE_ABORT_TEST          = 00000013     D
DS$K_TYPE_COMMAND_ERR         = 00000015     D
DS$K_TYPE_COMMAND_OUT         = 00000016     D
DS$K_TYPE_CRD_AUTOTEST        = 0000001A     D
DS$K_TYPE_DS_PROMPT           = 00000001     D
DS$K_TYPE_DS_START            = 0000001D     D
DS$K_TYPE_ERRDEV              = 00000008     D
DS$K_TYPE_ERRHARD             = 00000006     D
DS$K_TYPE_ERROR_BODY          = 00000009     D
DS$K_TYPE_ERROR_END           = 0000000A     D
DS$K_TYPE_ERRPREP             = 0000001B     D
DS$K_TYPE_ERRSOFT             = 00000007     D
DS$K_TYPE_ERRSUP              = 00000004     D
DS$K_TYPE_ERRSYS              = 00000005     D
DS$K_TYPE_ERR_HALT            = 0000000D     D
DS$K_TYPE_EXCEPTION           = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD      = 0000000B     D
DS$K_TYPE_FIRST_PASS          = 00000011     D
DS$K_TYPE_GENERAL             = 00000000     D
DS$K_TYPE_GENERAL_ERROR       = 00000003     D
DS$K_TYPE_NO_TESTS            = 00000012     D
DS$K_TYPE_PARAM_ERROR         = 0000001C     D
DS$K_TYPE_PROGRAM_END         = 00000010     D
DS$K_TYPE_PROGRAM_INFO        = 00000017     D
DS$K_TYPE_PROGRAM_START       = 0000000F     D
DS$K_TYPE_QIO_INVADP          = 00000024     D
DS$K_TYPE_QIO_NODRIVER        = 00000022     D
DS$K_TYPE_QIO_WRONGVER        = 00000023     D
DS$K_TYPE_SCRIPT_ECHO         = 00000021     D
DS$K_TYPE_SCRIPT_PNF          = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT       = 00000020     D
DS$K_TYPE_SCRIPT_SKIP         = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR      = 00000019     D
DS$K_TYPE_START_ERR           = 00000018     D
DS$K_TYPE_START_LIST          = 00000025     D
DS$K_TYPE_SUMMARY             = 0000000E     D
DS$K_TYPE_USER_PROMPT         = 00000002     D
DS$K_WARNING                  = 00000000     D
DS$LOAD                         ********   X   00
DS$_ARITH                     = 006600D0     D
DS$_ASBE                      = 00660118     D
DS$_BADLINK                   = 006600F0     D
DS$_BADTYPE                   = 006600E8     D
DS$_BIIC                      = 00660120     D
DS$_CHME                      = 006600A8     D
DS$_CHMK                      = 006600E0     D
DS$_DEVNAME                   = 00660108     D
DS$_ERROR                     = 00660002     D
DS$_FHWE                      = 00660068     D
DS$_FRAGBUF                   = 00660080     D
DS$_ICBUSY                    = 006600C8     D
DS$_ICERR                     = 006600C0     D
DS$_IHWE                      = 00660060     D
DS$_ILLCHAR                   = 00660018     D
DS$_ILLPAGCNT                 = 00660078     D
DS$_ILLUNIT                   = 00660100     D
DS$_INSFMEM                   = 00660050     D
DS$_IFL2HI                    = 006600B8     D
DS$_IVADDR                    = 00660040     D
DS$_IVVECT                    = 00660038     D
DS$_KRNLSTK                   = 00660090     D
DS$_LOGIC                     = 00660070     D
DS$_MCHK                      = 00660088     D
DS$_MMOFF                     = 00660058     D
DS$_NEEDUNIT                  = 006600F8     D
```

```
DS$_NODE                =  00660128     D
DS$_NOPCS               =  00660110     D
DS$_NORMAL              =  00660001     D
DS$_NOSUPPORT           =  006600B1     D
DS$_NOTDON              =  00660030     D
DS$_NOTIMP              =  006600B0     D
DS$_NULLSTR             =  00660010     D
DS$_OVERFLOW            =  00660008     D
DS$_POWER               =  00660098     D
DS$_PROGERR             =  00660020     D
DS$_SEVERE              =  00660004     D
DS$_TRANSL              =  006600A0     D
DS$_TRUNCATE            =  00660028     D
DS$_UNEXPINT            =  006600D8     D
DS$_VASFULL             =  00660048     D
DS$_WARNING             =  00660000     D
DSA$AL_APTMAIL             0000FE00     D
DSA$AT_APTTXT              0000FA00     D
DSA$GL_APTCOM              0000FE04     D
DSA$GL_DEVLEN              0000FE58     D
DSA$GL_ERRNO               0000FE44     D
DSA$GL_EVENT               0000FE48     D
DSA$GL_FLAGS               0000FE00     D
DSA$GL_MSGTYP              0000FE40     D
DSA$GL_PASSES              0000FE08     D
DSA$GL_PASSNO              0000FE54     D
DSA$GL_SECTNO              0000FE10     D
DSA$GL_SID                 0000FE14     D
DSA$GL_SUBTNO              0000FE4C     D
DSA$GL_TESTNO              0000FE50     D
DSA$GL_UNITS               0000FE0C     D
DSA$GQ_MSGPTR              0000FE68     D
DSA$GT_DEVNAM              0000FE5C     D
DSA$M_HALT              =  00000001     D
DSR$COMPLETION             ********   X    00
DSX$PRINT                  ********   X    00
DS_ERRSUP                  ********   X    00
DVR$B_CNUMVEC              0000000B     D
DVR$B_TYPE                 0000000A     D
DVR$C_LEN                  00000014     D
DVR$L_IDENT                00000010     D
DVR$W_DPT                  0000000E     D
DVR$W_GENERIC              0000000C     D
DVR$W_SIZE                 00000008     D
DVR$_IDENT              =  FFFF0001     D
DW750$K_LEN                00000032     D
DW780$B_BR                 00000033     D
DW780$B_TR                 00000032     D
DW780$K_LEN                00000034     D
DYN$C_ACB               =  00000002  G  D
DYN$C_ADP               =  00000001  G  D
DYN$C_AQB               =  00000003  G  D
DYN$C_BRDCST            =  0000001A  G  D
DYN$C_BUFIO             =  00000013  G  D
DYN$C_CEB               =  00000004  G  D
DYN$C_CRB               =  00000005  G  D
DYN$C_CXB               =  0000001B  G  D
DYN$C_DDB               =  00000006  G  D
DYN$C_DPT               =  0000001E  G  D
DYN$C_EXTGSD            =  00000028  G  D
DYN$C_FCB               =  00000007  G  D
DYN$C_FRK               =  00000008  G  D
DYN$C_GSD               =  00000015  G  D
DYN$C_IDB               =  00000009  G  D
DYN$C_IRP               =  0000000A  G  D
DYN$C_IRPE              =  0000002C  G  D
DYN$C_JIB               =  0000002F  G  D
DYN$C_JPB               =  0000001F  G  D
DYN$C_KFH               =  00000026  G  D
DYN$C_KFI               =  00000018  G  D
DYN$C_LOG               =  0000000B  G  D
DYN$C_MBX               =  0000002B  G  D
DYN$C_MTL               =  00000019  G  D
DYN$C_MVL               =  00000016  G  D
DYN$C_NDB               =  0000001C  G  D
DYN$C_NET               =  00000017  G  D
DYN$C_PBH               =  00000020  G  D
DYN$C_PCB               =  0000000C  G  D
DYN$C_PDB               =  00000021  G  D
DYN$C_PFL               =  00000023  G  D
DYN$C_PIB               =  00000022  G  D
DYN$C_PQB               =  0000000D  G  D
DYN$C_PTR               =  00000025  G  D
DYN$C_RBM               =  00000031  G  D
DYN$C_RVT               =  0000000E  G  D
DYN$C_RVX               =  00000027  G  D
DYN$C_SFT               =  00000024  G  D
DYN$C_SHB               =  0000002A  G  D
DYN$C_SHMCEB            =  0000002E  G  D
DYN$C_SHMGSD            =  00000029  G  D
DYN$C_SHRBUFIO          =  00000080  G  D
DYN$C_SLAVCEB           =  0000002D  G  D
DYN$C_SPECIAL           =  00000080  G  D
DYN$C_SSB               =  0000001D  G  D
DYN$C_TQE               =  0000000F  G  D
DYN$C_TWP               =  00000030  G  D
DYN$C_TYPAHD            =  00000014  G  D
DYN$C_UCB               =  00000010  G  D
DYN$C_VCA               =  00000032  G  D
DYN$C_VCB               =  00000011  G  D
DYN$C_WCB               =  00000012  G  D
EXE$ALONONPAGED            ********     X    00
EXE$DEANONPAGED            ********     X    00
EXE$FORKDSPTH              ********     X    00
EXE$GL_PWRDONE             00000004  RG  D    02
EXE$MAXACMODE              0000060C  RG  D    04
EXE$PWRTIMCHK              00000A25  RG  D    04
EXE$QIO                    00000000  RG  D    04
FILE                       FFFFFFF4      D
FILEDEF                    FFFFFFE4      D
FILEDESC                   FFFFFFEC      D
FILL_ACF                   000002B1  R   D    04
FIND_ADP                   00000A0D  R   D    04
HP$A_DEPENDENT             00000032      D
```

```
HP$A_DEVICE                     00000018     D
HP$A_DVA                        0000001C     D
HP$A_LINK                       00000020     D
HP$B_DRIVE                      0000000B     D
HP$B_FLAGS                      0000000A     D
HP$B_RH780_BR                   00000033     D
HP$B_RH780_TR                   00000032     D
HP$K_RH780_LEN                  00000034     D
HP$Q_DEVICE                     00000000     D
HP$T_DEVICE                     0000000C     D
HP$T_TYPE                       00000C026    D
HP$W_SIZE                       00000008     D
HP$W_VECTOR                     00000024     D
IDB$B_TYPE                      0000000A     D
IDB$C_LENGTH                    00000034     D
IDB$K_LENGTH                    00000034     D
IDB$L_ADP                       00000010     D
IDB$L_CSR                       00000000     D
IDB$L_OWNER                     00000004     D
IDB$L_UCBLST                    00000014     D
IDB$W_SIZE                      00000008     D
IDB$W_UNITS                     0000000C     D
INIT_CNTRL                      00000866  R  D  04
INIT_DB                         00000847  R  D  04
INIT_UNIT                       0000087A  R  D  04
INI_DRADP                       0000051B  R  D  04
INI_MBADP                       000003AB  R  D  04
INI_UBADP                       0000049F  R  D  04
IO$S_FCODE                    = 00000006     D
IO$TESTCSR_L                    ********     X   00
IO$TESTCSR_W                    ********     X   00
IO$V_FCODE                    = 00000000     D
IO$_WRITEVBLK                 = 00000030     D
IOC$GL_ADPLIST                  ********     X   00
IOC$GL_DEVLIST                  ********     X   00
IOC$GL_DPTLIST                  ********     X   00
IOC$INITDRV                     ********     X   00
IOC$IOPOST                      ********     X   00
IOC$REINITDRV                   ********     X   00
IOC$RTN                         ********     X   00
IOGEN$CNTRL_INI                 00000952  RG D  04
IOGEN$CONN_VEC                  ********     X   00
IOGEN$LOCK_IODB                 0000094A  R  D  04
IOGEN$UNLK_IODB                 00000945  R  D  04
LENGTH                          FFFFFFE0     D
LOAD$_ADDRESS                 = 00000010     D
LOAD$_DEFAULT                 = 00000008     D
LOAD$_FILE                    = 00000004     D
LOAD$_LENGTH                  = 0000000C     D
LOAD$_NARGS                   = 00000007     D
LOAD$_RETLEN                  = 00000014     D
LOAD$_RETREC                  = 00000018     D
LOAD$_VBN                     = 0000001C     D
LOAD_DRIVER                     00000165  R  D  04
LOAD_FLAGS                      00000004  R  D  03
LOAD_M_CRB                    = 00000002     D
LOAD_M_DDB                    = 00000001     D
LOAD_M_IDB                    = 00000004     D
LOAD_M_UCB                    = 00000008     D
LOAD_V_CRB                    = 00000001     D
LOAD_V_DDB                    = 00000000     D
LOAD_V_IDB                    = 00000002     D
LOAD_V_UCB                    = 00000003     D
LOC$DPT                         0000006E  R  D  04
LOC$PTDESC                      ********     X   00
LOCAL                           FFFFFFC0     D
MBA$INITIAL                     ********     X   00
MBA$INT                         ********     X   00
MBACRB                          0000048F  R  D  04
MBACRBLEN                     = 00000010     D
MBINT_DISP                      0000003C  R  D  02
NUMUBAVEC                     = 00000080     D
PR$_IPL                       = 00000012     D
PR$_TODR                      = 0000001B     D
PREADVBLK                       ********     X   00
PSL$S_PRVMOD                  = 00000002     D
PSL$V_PRVMOD                  = 00000016     D
QIO$ADDUNIT                     00000166  RG D  04
QIO$ALLOCATE                    000009BD  R  D  04
QIO$CLEANUP                     0000009F  RG D  04
QIO$DEALLOCATE                  000009E0  RG D  04
QIO$INITIALIZE                  0000004B  RG D  04
QIO$LOAD_DB                     00000638  R  D  04
QIO$L_ALLOCATE                  00000000  R  D  03
QIO$_ASTADR                   = 00000014     D
QIO$_ASTPRM                   = 00000018     D
QIO$_CHAN                     = 00000008     D
QIO$_EFN                      = 00000004     D
QIO$_FUNC                     = 0000000C     D
QIO$_IOSB                     = 00000010     D
QIO$_NARGS                    = 0000000C     D
QIO$_P1                       = 0000001C     D
QIO$_P2                       = 00000020     D
QIO$_P3                       = 00000024     D
QIO$_P4                       = 00000028     D
QIO$_P5                       = 0000002C     D
QIO$_P6                       = 00000030     D
QIOCLEAN_CPU                    ********     X   00
READVBLK                        ********     X   00
RH780$B_BR                      00000033     D
RH780$B_TR                      00000032     D
RH780$K_LEN                     00000034     D
SAVABS...                     = FFFFFFC0     D
SCAN$ALPHA                      ********     X   00
SCAN$SEARCHLIST                 ********     X   00
SCB$L_ACCESS                    00000020     D
SCB$L_ARITH                     00000034     D
SCB$L_BREAK                     0000002C     D
SCB$L_CHME                      00000044     D
SCB$L_CHMK                      00000040     D
SCB$L_CHMS                      00000048     D
SCB$L_CHMU                      0000004C     D
SCB$L_COMPAT                    00000030     D
SCB$L_KNLSTK                    00000008     D
```

QIO
Symbol table

SCB$L_MACHCHK	00000004		D			UCB$B_FIPL	0000000B		D
SCB$L_OPCCUS	00000014		D			UCB$B_LOCSRV	0000003C		D
SCB$L_OPCDEC	00000010		D			UCB$B_OFFNDX	00000094		D
SCB$L_POWER	0000000C		D			UCB$B_OFFRTC	00000095		D
SCB$L_RADRMOD	0000001C		D			UCB$B_REMSRV	0000003D		D
SCB$L_ROPRAND	00000018		D			UCB$B_SECTORS	0000003C		D
SCB$L_RXDB	000000F8		D			UCB$B_SLAVE	00000074		D
SCB$L_SFTLVL1	00000084		D			UCB$B_SPR	00000075		D
SCB$L_SFTLVL10	000000A8		D			UCB$B_STATE	00000052		D
SCB$L_SFTLVL11	000000AC		D			UCB$B_TRACKS	0000003D		D
SCB$L_SFTLVL12	000000B0		D			UCB$B_TT_CRFILL	0000009D		D
SCB$L_SFTLVL13	000000B4		D			UCB$B_TT_DECRF	000000A1		D
SCB$L_SFTLVL14	000000B8		D			UCB$B_TT_DELFF	000000A2		D
SCB$L_SFTLVL15	000000BC		D			UCB$B_TT_DESPEE	000000A0		D
SCB$L_SFTLVL2	00000088		D			UCB$B_TT_DETYPE	000000A4		D
SCB$L_SFTLVL3	0000008C		D			UCB$B_TT_LFFILL	0000009E		D
SCB$L_SFTLVL4	00000090		D			UCB$B_TT_SPEED	0000009C		D
SCB$L_SFTLVL5	00000094		D			UCB$B_TYPE	0000000A		D
SCB$L_SFTLVL6	00000098		D			UCB$B_VERTSZ	0000003F		D
SCB$L_SFTLVL7	0000009C		D			UCB$C_LENGTH	00000074		D
SCB$L_SFTLVL8	000000A0		D			UCB$C_MB_LENGTH	00000090		D
SCB$L_SFTLVL9	000000A4		D			UCB$C_TT_LENGTH	000000BC		D
SCB$L_TBIT	00000028		D			UCB$K_LENGTH	00000074		D
SCB$L_TIMER	000000C0		D			UCB$K_MB_LENGTH	00000090		D
SCB$L_TRANSL	00000024		D			UCB$K_TT_LENGTH	000000BC		D
SCB$L_TXDB	000000FC		D			UCB$L_AMB	00000054		D
SCB$L_ZERO	00000000		D			UCB$L_ASTQBL	00000010		D
SCB_BASE	********		X	00		UCB$L_ASTQFL	0000000C		D
SCB_IMAGE	********		X	00		UCB$L_CPID	0000005C		D
SIZ...	= 00000001		D			UCB$L_CRB	00000020		D
SS$_INSFMEM	********		X	00		UCB$L_DDB	00000024		D
SS$_IVCHAN	********		X	00		UCB$L_DEVCHAR	00000034		D
SS$_NORMAL	********		X	00		UCB$L_DEVDEPEND	0000003C		D
SYSG$_INVDPTINI	= 007C800A		D			UCB$L_DPC	00000080		D
SYSG$_INVVEC	= 007C802A		D			UCB$L_DUETIM	0000005C		D
SYSG$_VECINUSE	= 007C801A		D			UCB$L_DX_BFPNT	0000009C		D
T_ADAPTER_TYPE	000000C0	R	D	02		UCB$L_DX_BUF	00000098		D
T_INVADP	00000095	R	D	02		UCB$L_DX_RXDB	000000A0		D
T_KNOWN_DEV	000000C6	R	D	02		UCB$L_EMB	00000078		D
T_NODRIVER	00000046	R	D	02		UCB$L_FIRST	00000014		D
T_WRONGVER	00000068	R	D	02		UCB$L_FPC	0000000C		D
UBA$INITIAL	********		X	00		UCB$L_FQBL	00000004		D
UBA$UNEXINT	********		X	00		UCB$L_FQFL	00000000		D
UBADP_CPU	********		X	00		UCB$L_FR3	00000010		D
UBINTSZ	********		X	00		UCB$L_FR4	00000014		D
UBINT_DISP	00000044	RG	D	02		UCB$L_IOQBL	00000044		D
UCB$B_AMOD	00000053		D			UCB$L_IOQFL	00000040		D
UCB$B_CEX	00000077		D			UCB$L_IRP	0000004C		D
UCB$B_CM1	0000004A		D			UCB$L_LINK	0000002C		D
UCB$B_CM2	0000004B		D			UCB$L_LOGADR	00000064		D
UCB$B_DEVCLASS	00000038		D			UCB$L_MAXBLOCK	00000084		D
UCB$B_DEVTYPE	00000039		D			UCB$L_MB_MBX	0000007C		D
UCB$B_DIPL	00000052		D			UCB$L_MB_PORT	0000008C		D
UCB$B_DX_SCTCNT	000000A6		D			UCB$L_MB_RAST	00000078		D
UCB$B_ERTCNT	00000070		D			UCB$L_MB_SHB	00000080		D
UCB$B_ERTMAX	00000071		D			UCB$L_MB_WAST	00000074		D
UCB$B_FEX	00000076		D			UCB$L_MB_WIOQBL	00000088		D

```
UCB$L_MB_WIOQFL                 00000084     D
UCB$L_MEDIA                     0000008C     D
UCB$L_NT_DATSSB                 00000074     D
UCB$L_NT_INTSSB                 00000078     D
UCB$L_OPCNT                     00000060     D
UCB$L_OWNUIC                    0000001C     D
UCB$L_PID                       00000028     D
UCB$L_RQBL                      00000004     D
UCB$L_RQFL                      00000000     D
UCB$L_SVAPTE                    00000068     D
UCB$L_SVPN                      00000064     D
UCB$L_TT_DECHAR                 000000A8     D
UCB$L_TT_RDUE                   0000008C     D
UCB$L_TT_RTIMOU                 000000B8     D
UCB$L_VCB                       00000030     D
UCB$M_ONLINE                =   00000010     D
UCB$M_VALID                 =   00000800     D
UCB$T_PARTNER                   0000000C     D
UCB$W_BCNT                      0000006E     D
UCB$W_BCR                       00000096     D
UCB$W_BOFF                      0000006C     D
UCB$W_BUFQUO                    00000018     D
UCB$W_BYTESTOGO                 0000003E     D
UCB$W_CHARGE                    0000004A     D
UCB$W_CYLINDERS                 0000003E     D
UCB$W_DA                        0000008C     D
UCB$W_DC                        0000008E     D
UCB$W_DEVBUFSIZ                 0000003A     D
UCB$W_DEVSTS                    0000005A     D
UCB$W_DIRSEQ                    00000088     D
UCB$W_DSTADDR                   00000018     D
UCB$W_DX_BCR                    000000A4     D
UCB$W_ECT                       00000090     D
UCB$W_EC2                       00000092     D
UCB$W_ERRCNT                    00000072     D
UCB$W_FUNC                      0000007E     D
UCB$W_MB_SEED                   00000000     D
UCB$W_MSGCNT                    00000016     D
UCB$W_MSGMAX                    00000014     D
UCB$W_NT_CHAN                   0000007C     D
UCB$W_OFFSET                    0000008A     D
UCB$W_REFC                      00000050     D
UCB$W_SIZE                      00000008     D
UCB$W_SRCADDR                   0000001A     D
UCB$W_STS                       00000058     D
UCB$W_TT_DESIZE                 000000A5     D
UCB$W_UNIT                      00000048     D
UCB$W_VPROT                     0000001A     D
UCB_BLINK                       00000009 R   D    03
VEC$B_DATAPATH                  00000013     D
VEC$B_NUMREG                    00000012     D
VEC$C_LENGTH                    00000024     D
VEC$K_LENGTH                    00000024     D
VEC$L_ADP                       00000014     D
VEC$L_IDB                       00000008     D
VEC$L_INITIAL                   0000000C     D
VEC$L_START                     00000010     D
VEC$L_UNITDISC                  00000020     D
VEC$L_UNITINIT                  00000018     D
VEC$Q_DISPATCH                  00000000     D
VEC$W_MAPREG                    00000010     D
VMS$QIO                         ********   X   00
WRITEVBLK                       ********   X   00
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFF4 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000000F8 (248.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	0000000D (13.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000A36 (2614.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

QIO
Cross reference

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000007	236 (1)	211 (1) 222 (1) 236 (1)
$$T1	=00000020	236 (1)	211 (1) 222 (1) 236 (1)
$ER	=00000006	1112 (13)	#-1112 (13) #-382 (1) #-541 (4) #-768 (6) #-799 (6)
$MODULE	00000000-R	264 (1)	1112 (13) 382 (1) 541 (4) 768 (6) 799 (6)
$_KNOWN	=00000006	314 (1)	307 (1)
ACF	FFFFFFC0	608 (5)	613 (6)
ACF$B_AFLAG	0000000B		#-792 (6)
ACF$B_AUNIT	0000000A		#-1262 (15) #-1328 (15) #-1449 (16) #-775 (6)
ACF$B_CNUMVEC	00000013		#-1244 (15) #-1287 (15) #-789 (6) #-791 (6)
ACF$B_CUNIT	00000012		#-1203 (15) #-1322 (15) #-1327 (15) #-1423 (16) #-784 (6)
ACF$C_LENGTH	00000020		608 (5)
ACF$L_ADAPTER	00000000		#-1236 (15) #-1258 (15) #-1276 (15) #-1291 (15) #-1295 (15) #-1369 (15) #-770 (6)
ACF$L_CONFIGREG	00000004		#-771 (6)
ACF$L_CONTRLREG	0000000C		#-1277 (15) #-777 (6)
ACF$L_DEVNAME	00000014		1177 (15) #-1189 (15) #-1224 (15) 1226 (15) #-627 (6)
ACF$L_DRVNAME	00000018		#-1172 (15) #-1227 (15) 1229 (15) #-728 (6)
ACF$W_AVECTOR	00000008		#-773 (6)
ACF$W_CVECTOR	00000010		#-1283 (15) #-779 (6) #-782 (6)
ACF$W_MAXUNITS	0000001C		#-786 (6)
ADP$B_TYPE	0000000A		#-1031 (11) #-877 (8) #-930 (9)
ADP$C_DRADPLEN	00000014		#-988 (10)
ADP$C_MBAADPLEN	00000014		#-834 (7)
ADP$C_UBAADPLEN	00000070		#-921 (9)
ADP$L_CRB	00000010		#-1035 (11) #-1237 (15) #-1259 (15) #-881 (8)
ADP$L_CSR	00000000		#-1032 (11) #-1370 (15) #-1495 (17) #-1498 (17) #-1602 (19) #-878 (8) #-932 (9) 934 (9)
ADP$L_DPQFL	00000014		1583 (19) #-1584 (19) #-1588 (19) 1599 (19) #-1600 (19) #-943 (9) 937 (9)
ADP$L_LINK	00000004		
ADP$L_MRQFL	0000001C		#-1296 (15) #-947 (9)
ADP$L_VECTOR	00000010		#-1034 (11) #-880 (8) #-931 (9) #-941 (9)
ADP$W_ADPTYPE	0000000E		#-838 (7) #-929 (9) #-992 (10)
ADP$W_MRBITMAP	00000026		#-1033 (11) #-879 (8) #-933 (9)
ADP$W_SIZE	00000008		#-1039 (11) #-885 (8) #-945 (9)
ADP$W_TR	0000000C		
ADPLINK	000009F6-R	1582 (19)	
ALLOC$_ACMODE	=00000010	211 (1)	
ALLOC$_DEVNAM	=00000004	211 (1)	
ALLOC$_NARGS	=00000004	211 (1)	
ALLOC$_PHYBUF	=0000000C	211 (1)	
ALLOC$_PHYLEN	=00000008	211 (1)	
AT$_DR	=00000002		#-1034 (11) 304 (1)
AT$_MBA	=00000000		#-1513 (17) 300 (1) 302 (1) #-767 (6) #-880 (8)
AT$_UBA	=00000001		#-1234 (15) #-1293 (15) #-1367 (15) #-1487 (17)

```
                                                        299     (1)       301     (1)       303     (1)     #-931     (9)

AUNKFUNC                        00000008-R      269     (1)       382     (1)
BIT...                         =00000004        333     (1)       227     (1)       228     (1)       238     (1)       333     (1)
BUG$CHECK                       0000061B-R      1109    (13)  #-1113    (13)
CMK$_                          =00000004        228     (1)
CMK$_ASTEXIT                   =00000000        228     (1)
CMK$_CANTIM                    =0000000B        228     (1)
CMK$_HIBER                     =00000007        228     (1)
CMK$_INITSCB                   =00000008        228     (1)
CMK$_LAST                      =0000000E        228     (1)
CMK$_MMENABLE                  =00000003        228     (1)
CMK$_RCONSOLE                  =00000001        228     (1)
CMK$_REFRESH                   =00000005        228     (1)
CMK$_SCHDWK                    =00000006        228     (1)
CMK$_SETIMR                    =0000000C        228     (1)
CMK$_SETIPL                    =00000009        228     (1)
CMK$_SETPRT                    =0000000D        228     (1)
CMK$_SGIPR                     =0000000A        228     (1)
CMK$_TCONSOLE                  =00000002        228     (1)
CRB$B_TYPE                      0000000A                    #-1020    (11)  #-1254    (15)  #-866     (8)
CRB$C_LENGTH                    00000038                    #-822     (7)   #-976     (10)
CRB$K_LENGTH                    00000038                    #-1247    (15)
CRB$L_INTD                      00000014                      1007    (11)    1016    (11)    1017    (11)    1018    (11)
                                                              1019    (11)  #-1028    (11)  #-1029    (11)  #-1036    (11)
                                                            #-1202    (15)  #-1238    (15)  #-1261    (15)    1263    (15)
                                                              1288    (15)  #-1365    (15)  #-1382    (15)  #-1448    (16)
                                                            #-1492    (17)  #-1503    (17)  #-1518    (17)  #-1520    (17)
                                                            #-523     (4)     853     (8)     862     (8)     863     (8)
                                                              864     (8)     865     (8)   #-874     (8)   #-875     (8)
                                                            #-882     (8)
CRB$L_LINK                      00000010                    #-1260    (15)  #-1363    (15)  #-1447    (16)  #-506     (4)
CRB$L_WQBL                      00000004                    #-1022    (11)  #-1256    (15)  #-868     (8)
CRB$L_WQFL                      00000000                      1021    (11)    1022    (11)    1255    (15)    1256    (15)
                                                              867     (8)     868     (8)
CRB$W_REFC                      0000000C                    #-1023    (11)  #-1321    (15)  #-1422    (16)  #-869     (8)
CRB$W_SIZE                      00000008                    #-1253    (15)  #-826     (7)   #-980     (10)
CREATE_CRB                      00000710-R      1243    (15)  #-1233    (15)  #-1235    (15)
CREATE_DDB                      000006B4-R      1212    (15)  #-1193    (15)
CREATE_IDB                      0000075C-R      1267    (15)  #-1257    (15)
CREATE_UCB                      000007DA-R      1309    (15)  #-1208    (15)  #-1239    (15)
CREATE_VEC                      00000786-R      1282    (15)
CTL$GW_CHINDX                   0000003A-R      276     (1)
CTL$GW_NMIOCH                   00000038-R      273     (1)
DB_ERROR                        000008D8-R      1415    (16)  #-1185    (15)  #-1216    (15)  #-1250    (15)  #-1271    (15)
                                                            #-1286    (15)  #-1300    (15)  #-1313    (15)  #-1349    (15)
DB_EXIT                         000008D1-R      1391    (15)  #-1205    (15)
DDB$B_TYPE                      0000000A                    #-1222    (15)
DDB$K_LENGTH                    00000034                    #-1213    (15)
DDB$L_DDT                       0000000C                    #-1376    (15)
DDB$L_LINK                      00000000                    #-1192    (15)  #-1194    (15)    1218    (15)  #-1219    (15)
                                                            #-517     (4)
DDB$L_UCB                       00000004                    #-1200    (15)    1330    (15)  #-1490    (17)  #-1516    (17)
                                                            #-495     (4)     509     (4)
DDB$T_DRVNAME                   00000024                      1229    (15)
DDB$T_NAME                      00000014                      1195    (15)    1226    (15)
DDB$W_SIZE                      00000008                    #-1221    (15)
DDB_BLINK                       00000005-R      334     (1)   #-1218    (15)  #-1458    (16)
```

```
DDT$L_UNITINIT                  00000018                        #-1377    (15)
DIR...                         =FFFFFFFF      608     (5)        608     (5)
DPT$B_ADPTYPE                   0000000C                        #-1234    (15)  #-1293    (15)  #-1367    (15)  #-1487    (17)
                                                                #-1513    (17)  #-750     (6)
DPT$B_FLAGS                     0000000D                          1233    (15)    1257    (15)    1446    (16)
DPT$B_REFC                      0000000B                        #-1223    (15)  #-1457    (16)  #-535     (4)  #-795     (6)
                                                                #-797     (6)
DPT$L_FLINK                     00000000                        #-433     (3)   #-532     (4)  #-649     (6)
DPT$T_NAME                      0000001E                          438     (3)   #-652     (6)    727     (6)
DPT$V_SUBCNTRL                 =00000000                        #-1233    (15)  #-1257    (15)  #-1446    (16)
DPT$W_MAXUNITS                  00000016                        #-785     (6)
DPT$W_SIZE                      00000008                        #-720     (6)
DPT$W_UCBSIZE                   0000000E                        #-1310    (15)
DR$INITIAL                      00000000-XR                       1029    (11)    1040    (11)    1051    (11)     250    (1)
DR$INT                          00000000-XR                       1049    (11)     250    (1)
DRACRB                          000005FC-R     1047     (11)      1007    (11)    1052    (11)
DRACRBLEN                      =00000010       1052     (11)  #-1007    (11)
DS$GPHARD                       00000000-XR                       247     (1)
DS$K_CCB                        00000000-XR                       253     (1)     274     (1)     277     (1)
DS$K_ERROR                     =00000002        238     (1)
DS$K_NORMAL                    =00000001        238     (1)
DS$K_PRINTB                    =00000002        227     (1)
DS$K_PRINTF                    =00000001        227     (1)  #-1181    (15)  #-680     (6)  #-708     (6)  #-755     (6)
DS$K_PRINTI                    =00000000        227     (1)
DS$K_PRINTX                    =00000003        227     (1)
DS$K_SEVERE                    =00000004        238     (1)
DS$K_SUBSYS                    =00000066        238     (1)     238     (1)
DS$K_TYPE_ABORT_PROGRAM        =00000014        227     (1)
DS$K_TYPE_ABORT_TEST           =00000013        227     (1)
DS$K_TYPE_COMMAND_ERR          =00000015        227     (1)
DS$K_TYPE_COMMAND_OUT          =00000016        227     (1)
DS$K_TYPE_CRD_AUTOTEST         =0000001A        227     (1)
DS$K_TYPE_DS_PROMPT            =00000001        227     (1)
DS$K_TYPE_DS_START             =0000001D        227     (1)
DS$K_TYPE_ERRDEV               =00000008        227     (1)
DS$K_TYPE_ERRHARD              =00000006        227     (1)
DS$K_TYPE_ERROR_BODY           =00000009        227     (1)
DS$K_TYPE_ERROR_END            =0000000A        227     (1)
DS$K_TYPE_ERRPREP              =0000001B        227     (1)
DS$K_TYPE_ERRSOFT              =00000007        227     (1)
DS$K_TYPE_ERRSUP               =00000004        227     (1)
DS$K_TYPE_ERRSYS               =00000005        227     (1)
DS$K_TYPE_ERR_HALT             =0000000D        227     (1)
DS$K_TYPE_EXCEPTION            =0000000C        227     (1)
DS$K_TYPE_EXCEPTION_HEAD       =0000000B        227     (1)
DS$K_TYPE_FIRST_PASS           =00000011        227     (1)
DS$K_TYPE_GENERAL              =00000000        227     (1)
DS$K_TYPE_GENERAL_ERROR        =00000003        227     (1)
DS$K_TYPE_NO_TESTS             =00000012        227     (1)
DS$K_TYPE_PARAM_ERROR          =0000001C        227     (1)
DS$K_TYPE_PROGRAM_END          =00000010        227     (1)
DS$K_TYPE_PROGRAM_INFO         =00000017        227     (1)
DS$K_TYPE_PROGRAM_START        =0000000F        227     (1)
DS$K_TYPE_QIO_INVADP           =00000024        227     (1)  #-756     (6)
DS$K_TYPE_QIO_NODRIVER         =00000022        227     (1)  #-1182    (15)  #-681     (6)
DS$K_TYPE_QIO_WRONGVER         =00000023        227     (1)  #-709     (6)
DS$K_TYPE_SCRIPT_ECHO          =00000021        227     (1)
```

```
DS$K_TYPE_SCRIPT_PNF            =0000001E      227     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      227     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      227     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      227     (1)
DS$K_TYPE_START_ERR             =00000018      227     (1)
DS$K_TYPE_START_LIST            =00000025      227     (1)
DS$K_TYPE_SUMMARY               =0000000E      227     (1)
DS$K_TYPE_USER_PROMPT           =00000002      227     (1)
DS$K_WARNING                    =00000000      238     (1)
DS$LOAD                          00000000-XR                    248     (1)     674     (6)     695     (6)
DS$_ARITH                       =006600D0      238     (1)
DS$_ASBE                        =00660118      238     (1)
DS$_BADLINK                     =006600F0      238     (1)
DS$_BADTYPE                     =006600E8      238     (1)
DS$_BIIC                        =00660120      238     (1)
DS$_CHME                        =006600A8      238     (1)
DS$_CHMK                        =006600E0      238     (1)
DS$_DEVNAME                     =00660108      238     (1)
DS$_ERROR                       =00660002      238     (1)
DS$_FHWE                        =00660068      238     (1)
DS$_FRAGBUF                     =00660080      238     (1)
DS$_ICBUSY                      =006600C8      238     (1)
DS$_ICERR                       =006600C0      238     (1)
DS$_IHWE                        =00660060      238     (1)
DS$_ILLCHAR                     =00660018      238     (1)
DS$_ILLPAGCNT                   =00660078      238     (1)
DS$_ILLUNIT                     =00660100      238     (1)
DS$_INSFMEM                     =00660050      238     (1)
DS$_IPL2HI                      =006600B8      238     (1)
DS$_IVADDR                      =00660040      238     (1)
DS$_IVVECT                      =00660038      238     (1)
DS$_KRNLSTK                     =00660090      238     (1)
DS$_LOGIC                       =00660070      238     (1)
DS$_MCHK                        =00660088      238     (1)
DS$_MMOFF                       =00660058      238     (1)
DS$_NEEDUNIT                    =006600F8      238     (1)
DS$_NODE                        =00660128      238     (1)
DS$_NOPCS                       =00660110      238     (1)
DS$_NORMAL                      =00660001      238     (1)
DS$_NOSUPPORT                   =006600B1      238     (1)     #-641   (6)
DS$_NOTDON                      =00660030      238     (1)
DS$_NOTIMP                      =006600B0      238     (1)       238   (1)
DS$_NULLSTR                     =00660010      238     (1)
DS$_OVERFLOW                    =00660008      238     (1)
DS$_POWER                       =00660098      238     (1)
DS$_PROGERR                     =00660020      238     (1)
DS$_SEVERE                      =00660004      238     (1)
DS$_TRANSL                      =006600A0      238     (1)
DS$_TRUNCATE                    =00660028      238     (1)
DS$_UNEXPINT                    =006600D8      238     (1)
DS$_VASFULL                     =00660048      238     (1)
DS$_WARNING                     =00660000      238     (1)       238   (1)
DSA$GL_FLAGS                     0000FE00                       #-1111  (13)
DSA$M_HALT                      =00000001                       #-1111  (13)
DSR$COMPLETION                   00000000-XR                      248   (1)     689     (6)     697     (6)
DSX$PRINT                        00000000-XR                     1183   (15)    247     (1)     682     (6)     710     (6)
                                                                 757   (6)
```

```
DS_ERRSUP                       00000000-XR                    1112 (13)     247 (1)     382 (1)     541 (4)
                                                                768 (6)     799 (6)
DVR$B_CNUMVEC                   0000000B                     #-788 (6)
DVR$C_LEN                       00000014                       719 (6)     787 (6)
DVR$L_IDENT                     00000010                     #-702 (6)   #-705 (6)
DVR$W_SIZE                      00000008                     #-700 (6)
DVR$_IDENT                      =FFFF0001                    #-701 (6)   #-704 (6)
DYN$C_ADP                       =00000001                    #-1031 (11) #-877 (8)   #-930 (9)
DYN$C_CRB                       =00000005                    #-1020 (11) #-1254 (15) #-866 (8)
DYN$C_DDB                       =00000006                    #-1222 (15)
DYN$C_IDB                       =00000009                    #-1025 (11) #-1275 (15) #-871 (8)
DYN$C_UCB                       =00000010                    #-1317 (15)
EXE$ALONONPAGED                 00000000-XR                    1547 (18)     245 (1)     687 (6)
EXE$DEANONPAGED                 00000000-XR                    1569 (18)     245 (1)     715 (6)
EXE$FORKDSPTH                   00000000-XR                     257 (1)
EXE$GL_PWRDONE                  00000004-R    266   (1)      #-1626 (19)
EXE$MAXACMODE                   0000060C-R    1073  (12)
EXE$PWRTIMCHK                   00000A25-R    1623  (19)
EXE$QIO                         00000000-R    363   (1)
FILE                            FFFFFFF4      608   (5)      #-662 (6)   #-663 (6)   #-664 (6)     666 (6)
FILEDEF                         FFFFFFE4      608   (5)      #-667 (6)     671 (6)
FILEDESC                        FFFFFFEC      608   (5)      #-665 (6)   #-666 (6)     672 (6)     678 (6)
                                                               706 (6)
FILL_ACF                        000002B1-R    726   (6)      #-654 (6)
FIND_ADP                        00000A0D-R    1597  (19)     #-763 (6)
HP$A_DEVICE                     00000018                     #-761 (6)   #-776 (6)
HP$A_LINK                       00000020                     #-639 (6)   #-731 (6)   #-734 (6)
HP$B_DRIVE                      0000000B                     #-774 (6)   #-783 (6)
HP$Q_DEVICE                     00000000                     #-618 (6)   #-619 (6)     677 (6)     752 (6)
HP$T_TYPE                       00000026                       630 (6)   #-739 (6)     744 (6)     753 (6)
HP$W_VECTOR                     00000024                     #-772 (6)   #-778 (6)   #-781 (6)
IDB$B_TYPE                      0000000A                     #-1025 (11) #-1275 (15) #-871 (8)
IDB$C_LENGTH                    00000034                     #-828 (7)   #-982 (10)
IDB$K_LENGTH                    00000034                     #-1268 (15)
IDB$L_ADP                       00000010                     #-1037 (11) #-1276 (15) #-1494 (17) #-1497 (17)
                                                             #-883 (8)
IDB$L_CSR                       00000000                     #-1027 (11) #-1277 (15) #-1361 (15) #-1366 (15)
                                                             #-1493 (17) #-1519 (17) #-873 (8)
IDB$L_UCBLST                    00000014                     #-1263 (15) #-1323 (15) #-1424 (16) #-1450 (16)
IDB$W_SIZE                      00000008                     #-1274 (15) #-832 (7)   #-986 (10)
IDB$W_UNITS                     0000000C                     #-1026 (11) #-1278 (15) #-872 (8)
INIT_CNTRL                      00000866-R    1353  (15)     #-1347 (15)
INIT_DB                         00000847-R    1342  (15)
INIT_UNIT                       0000087A-R    1359  (15)     #-1355 (15)
INI_DRADP                       00000518-R    972   (10)       767 (6)
INI_MBADP                       000003AB-R    818   (7)        767 (6)
INI_UBADP                       0000049F-R    918   (9)        767 (6)
IO$S_FCODE                      =00000006                    #-370 (1)
IO$TESTCSR_L                    00000000-XR                  #-1372 (15)     246 (1)
IO$TESTCSR_W                    00000000-XR                  #-1500 (17)     246 (1)
IO$V_FCODE                      =00000000                    #-370 (1)
IO$_WRITEVBLK                   =00000030                    #-380 (1)
IOC$GL_ADPLIST                  00000000-XR                    1583 (19)    1599 (19)     244 (1)     #-420 (2)
                                                                482 (4)
IOC$GL_DEVLIST                  00000000-XR                    1191 (15)     244 (1)     #-419 (2)     #-492 (4)
                                                             #-517 (4)
IOC$GL_DPTLIST                  00000000-XR                     244 (1)      415 (2)      430 (3)      530 (4)
```

```
                                                      647     (6)     721     (6)
IOC$INITDRV                    00000000-XR                    1345    (15)     256     (1)
IOC$IOPOST                     00000000-XR                     257    (1)
IOC$REINITDRV                  00000000-XR                     257    (1)
IOC$RTN                        00000000-XR                   #-1380   (15)     259     (1)
IOGEN$CNTRL_INI                00000952-R    1486    (17)   #-1357   (15)
IOGEN$CONN_VEC                 00000000-XR                   #-1302   (15)     253     (1)
IOGEN$LOCK_IODB                0000094A-R    1470    (16)   #-1168   (15)
IOGEN$UNLK_IODB                0000094E-R    1473    (16)   #-1392   (15)   #-1462   (16)
LENGTH                         FFFFFFE0       608     (5)     669    (6)   #-686    (6)
LOAD$_ADDRESS                  =00000010      236     (1)   #-694    (6)
LOAD$_DEFAULT                  =00000008      236     (1)
LOAD$_FILE                     =00000004      236     (1)
LOAD$_LENGTH                   =0000000C      236     (1)   #-693    (6)
LOAD$_NARGS                    =00000007      236     (1)
LOAD$_RETLEN                   =00000014      236     (1)
LOAD$_RETREC                   =00000018      236     (1)
LOAD$_VBN                      =0000001C      236     (1)
LOAD_DRIVER                    000001E5-R     661     (6)   #-651    (6)
LOAD_FLAGS                     00000004-R     326     (1)   #-1167   (15)   #-1217   (15)   #-1251   (15)   #-1272   (15)
                                                            #-1314   (15)     1355   (15)     1420   (16)     1439   (16)
                                                              1445   (16)     1456   (16)
LOAD_M_CRB                     =00000002      333     (1)   #-1251   (15)
LOAD_M_DDB                     =00000001      333     (1)   #-1217   (15)
LOAD_M_IDB                     =00000004      333     (1)   #-1272   (15)
LOAD_M_UCB                     =00000008      333     (1)   #-1314   (15)
LOAD_V_CRB                     =00000001      333     (1)   #-1355   (15)   #-1445   (16)
LOAD_V_DDB                     =00000000      333     (1)   #-1456   (16)
LOAD_V_IDB                     =00000002      333     (1)   #-1439   (16)
LOAD_V_UCB                     =00000003      333     (1)   #-1420   (16)
LOC$DPT                        0000006E-R     428     (3)   #-1174   (15)
LOC$PTDESC                     00000000-XR                    258    (1)     632    (6)
LOCAL                          FFFFFFC0       608     (5)     612    (6)
MBA$INITIAL                    00000000-XR                    249    (1)     875    (8)     886    (8)     897    (8)
MBA$INT                        00000000-XR                    249    (1)     895    (8)
MBACRB                         0000048F-R     893     (8)     853    (8)     898    (8)
MBACRBLEN                      =00000010      898     (8)   #-853    (8)
MBINT_DISP                     0000003C-R     279     (1)   #-1289   (15)
NUMUBAVEC                      =00000080      916     (9)   #-921    (9)   #-948    (9)
PR$_IPL                        =00000012                    #-1005   (11)   #-1041   (11)   #-1354   (15)   #-1387   (15)
                                                            #-1471   (16)   #-1474   (16)   #-480    (4)   #-538    (4)
                                                            #-851    (8)   #-887    (8)   #-927    (9)   #-951    (9)
PR$_TODR                       =0000001B                    #-1625   (19)
PREADVBLK                      00000000-XR                    254    (1)
PSL$S_PRVMOD                   =00000002                    #-1075   (12)   #-1077   (12)
PSL$V_IS                       =0000001A      228     (1)
PSL$V_PRVMOD                   =00000016                    #-1075   (12)   #-1077   (12)
QIO$ADDUNIT                    00000166-R     610     (6)
QIO$ALLOCATE                   000009BD-R    1545    (18)   #-1214   (15)   #-1248   (15)   #-1269   (15)   #-1311   (15)
                                                            #-823    (7)   #-829    (7)   #-835    (7)   #-923    (9)
                                                            #-977    (10)  #-983    (10)  #-989    (10)
QIO$CLEANUP                    0000009F-R     476     (4)
QIO$DEALLOCATE                 000009E0-R    1565    (18)   #-1433   (16)   #-513    (4)   #-518    (4)   #-524    (4)
                                                            #-526    (4)   #-843    (7)   #-845    (7)   #-997    (10)
                                                            #-999    (10)
QIO$INITIALIZE                 0000004B-R     413     (2)
QIO$LOAD_DB                    00000638-R    1164    (15)     794    (6)
```

```
QIO$L_ALLOCATE                  00000000-R    323  (1)     #-1555  (18)   #-1568  (18)   #-421   (2)    #-539   (4)
QIO$_ASTADR                    =00000014      222  (1)
QIO$_ASTPRM                    =00000018      222  (1)
QIO$_CHAN                      =00000008      222  (1)     #-365   (1)
QIO$_EFN                       =00000004      222  (1)
QIO$_FUNC                      =0000000C      222  (1)       37.   (1)
QIO$_IOSB                      =00000010      222  (1)
QIO$_NARGS                     =0000000C      222  (1)
QIO$_P1                        =0000001C      222  (1)
QIO$_P2                        =00000020      222  (1)
QIO$_P3                        =00000024      222  (1)
QIO$_P4                        =00000028      222  (1)
QIO$_P5                        =0000002C      222  (1)
QIO$_P6                        =00000030      222  (1)
QIOCLEAN_CPU                    00000000-XR                   258  (1)      484  (4)
READVBLK                        00000000-XR                   254  (1)      386  (1)
SAVABS...                      =FFFFFFC0      608  (5)
SCAN$ALPHA                      00000000-XR                   256  (1)      620  (6)
SCAN$SEARCHLIST                 00000000-XR                   256  (1)      747  (6)
SCB$L_SFTLVL4                   00000090                    #-487  (4)   #-488  (4)
SCB$L_SFTLVL8                   000000A0                    #-485  (4)   #-486  (4)
SCB_BASE                        00000000-XR                  1013  (11)     1015  (11)     252  (1)      478  (4)
                                                              614  (6)      859  (8)      861  (8)
SCB_IMAGE                       00000000-XR                   252  (1)      479  (4)
SIZ...                         =00000001      333  (1)       333  (1)
SS$_INSFMEM                     00000000-XR                 #-1001  (10)     255  (1)    #-847   (7)
SS$_IVCHAN                      00000000-XR                   255  (1)
SS$_NORMAL                      00000000-XR                 #-1042  (11)   #-1604  (19)     255  (1)    #-442   (3)
                                                            #-888   (8)
SYSG$_INVDPTINI                =007C800A                    #-1348  (15)
SYSG$_INVVEC                   =007C802A                    #-1285  (15)
SYSG$_VECINUSE                 =007C801A                    #-1299  (15)
T_ADAPTER_TYPE                  000000C0-R    298  (1)     #-749   (6)
T_INVADP                        00000095-R    292  (1)       754   (6)
T_KNOWN_DEV                     000000C6-R    306  (1)       314   (1)      746   (6)
T_NODRIVER                      00000046-R    286  (1)      1180   (15)     679   (6)
T_WRONGVER                      00000068-R    289  (1)       707   (6)
UBA$INITIAL                     00000000-XR                   251   (1)   #-944   (9)
UBA$UNEXINT                     00000000-XR                 #-1297  (15)     249   (1)      949   (9)
UBADP_CPU                       00000000-XR                   252   (1)   #-946   (9)
UBINTSZ                         00000000-XR                   251   (1)   #-921   (9)
UBINT_DISP                      00000044-R    283  (1)
UCB$B_SLAVE                     00000074                    #-1328  (15)
UCB$B_TYPE                      0000000A                    #-1317  (15)
UCB$L_ASTQBL                    00000010                    #-1319  (15)
UCB$L_ASTQFL                    0000000C                     1318   (15)     1319  (15)
UCB$L_CRB                       00000020                    #-1201  (15)   #-1320  (15)  #-1491  (17)  #-1517  (17)
                                                            #-496   (4)
UCB$L_DDB                       00000024                    #-1324  (15)
UCB$L_IOQBL                     00000044                    #-1326  (15)
UCB$L_IOQFL                     00000040                     1325   (15)     1326  (15)
UCB$L_LINK                      0000002C                    #-1206  (15)     1333  (15)     509   (4)    #-510   (4)
UCB$M_ONLINE                   =00000010                    #-1374  (15)
UCB$M_VALID                    =00000800                    #-1346  (15)
UCB$W_SIZE                      00000008                    #-1316  (15)
UCB$W_STS                       00000058                    #-1346  (15)   #-1374  (15)
UCB$W_UNIT                      00000048                    #-1203  (15)   #-1327  (15)
```

```
UCB_BLINK                       00000009-R  337     (1)     #-1336  (15)    #-1421  (16)
VEC$K_LENGTH                    00000024                    #-1246  (15)    #-1304  (15)
VEC$L_ADP                       00000014                    #-1036  (11)    #-1291  (15)    #-882   (8)
VEC$L_IDB                       00000008                    #-1028  (11)    #-1202  (15)    #-1238  (15)    #-1261  (15)
                                                            #-1292  (15)    #-1365  (15)    #-1448  (16)    #-1492  (17)
                                                            #-1518  (17)    #-523   (4)     #-874   (8)
VEC$L_INITIAL                   0000000C                    #-1029  (11)    #-1503  (17)    #-1520  (17)    #-875   (8)
VEC$L_UNITINIT                  00000018                    #-1382  (15)
VEC$Q_DISPATCH                  00000000                    #-1289  (15)
VMS$QIO                         00000000-XR                   253   (1)       367   (1)
WRITEVBLK                       00000000-XR                   254   (1)       384   (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACFDEF	2	209 (1)	209 (1)
$ADPDEF	2	210 (1)	210 (1)
$ALLOCDEF	1	211 (1)	211 (1)
$CCBDEF	1	212 (1)	212 (1)
$CRBDEF	1	213 (1)	213 (1)
$DCDEF	3	214 (1)	214 (1)
$DDBDEF	1	215 (1)	215 (1)
$DDTDEF	1	216 (1)	216 (1)
$DEF	1	238 (1)	
$DEFINI	1	203 (1)	203 (1) 209 (1) 210 (1) 212 (1) 213 (1)
			214 (1) 215 (1) 216 (1) 217 (1) 218 (1)
			219 (1) 220 (1) 221 (1) 223 (1) 224 (1)
			225 (1) 226 (1) 230 (1) 231 (1) 232 (1)
			233 (1) 234 (1) 235 (1) 237 (1) 239 (1)
$DPTDEF	2	217 (1)	217 (1)
$DS_DR780_DEF	1	234 (1)	234 (1)
$DS_DSADEF	5	239 (1)	239 (1)
$DS_DSDEF	2	238 (1)	238 (1)
$DS_DW750_DEF	1	237 (1)	237 (1)
$DS_DW780_DEF	1	233 (1)	233 (1)
$DS_HPODEF	2	231 (1)	231 (1)
$DS_LOAD_DEF	1	236 (1)	236 (1)
$DS_QIODVR_DEF	2	230 (1)	230 (1)
$DS_RH780_DEF	1	232 (1)	232 (1)
$DS_SCBDEF	3	235 (1)	235 (1)
$DS_TYPEDEF	4	227 (1)	227 (1)
$DYNDEF	2	218 (1)	218 (1)
$EQU	1	238 (1)	227 (1) 228 (1) 238 (1)
$EQULS1	1	238 (1)	227 (1) 228 (1) 238 (1)
$EQULST	1		227 (1) 228 (1) 238 (1)
$GBLINI	2		227 (1) 228 (1) 238 (1)
$IDBDEF	1	219 (1)	219 (1)
$IODEF	17	220 (1)	220 (1)
$OFFDEF	1	211 (1)	211 (1) 222 (1) 236 (1)
$OFFSET	2	602 (5)	602 (5)
$OFFST1	1	608 (5	608 (5)
$PRDEF	4	117 (1.	203 (1)
$PSLDEF	2	221 (1)	221 (1)
$PUSHADR	1	382 (1)	1112 (13) 382 (1) 541 (4) 768 (6) 799 (6)
$QIODEF	1	222 (1)	222 (1)
$SYSGMSGDEF	5	223 (1)	223 (1)
$UBADEF	6	224 (1)	224 (1)
$UCBDEF	10	225 (1)	225 (1)
$VECDEF	2	226 (1)	226 (1)
$VIELD1	1	238 (1)	333 (1)
CASE	1	372 (1)	372 (1) 766 '5)
CMKDEF	1	228 (1)	228 (1)
DSBINT	1	480 (4)	1005 (11) 1354 (15) 480 (4) 851 (8) 927 (9)
ENBINT	1	538 (4)	1041 (11) 1387 (15) 538 (4) 887 (8) 951 (9)
ERRSUP_S	1	382 (1)	1112 (13) 382 (1) 541 (4) 768 (6) 799 (6)

MODNAM	1	264	(1)	264	(1)		
SETIPL	1	1471	(16)	1471	(16)	1474	(16)
_VIELD	1	328	(1)	328	(1)		

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.11	00:00:00.28
Command processing	147	00:00:00.83	00:00:01.81
Pass 1	1793	00:00:30.44	00:00:43.11
Symbol table sort	26	00:00:02.37	00:00:02.61
Pass 2	677	00:00:06.17	00:00:11.42
Symbol table output	51	00:00:00.48	00:00:01.71
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	148	00:00:01.88	00:00:03.46
Assembler run totals	2886	00:00:42.33	00:01:04.44

The working set limit was 1000 pages.
168454 bytes (330 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1494 non-local and 104 local symbols.
1630 source lines were read in Pass 1, producing 0 object records in Pass 2.
260 pages of virtual memory were used to define 56 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	24
DRB1:[DS.WORK]DS.MLB;218	3
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	6
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	13
TOTALS (all libraries)	46

1896 GETS were required to define 46 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) QIO/UPDA=(QIO.UPD,QIO.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DIAG/
```

Table of contents

(1) 72.2 ONE PARAMETER FUNCTION PROCESSING
(1) 99.9 ZERO PARAMETER FUNCTION PROCESSING
(1) 158.6 READ AND WRITE FUNCTION PROCESSING
(1) 191.23 READ AND WRITE FUNCTION BUFFER CHECK AND LOCK ROUTINES
(1) 293 READ AND WRITE BUFFER CHECK AND LOCK AND RETURN ROUTINES
(1) 394 CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
(1) 430 CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
(1) 469 CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
(1) 516 CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
(1) 569 SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
(1) 609 SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS
(1) 648 SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
(1) 686 CARRIAGE CONTROL INTERPRETATION

```
                                0000     .1          .TITLE  QIOFDT  *** QIOFDT function dispatch routines
                                0000     .2          .IDENT  /05-03/
                            00000000     .3          .PSECT  SEP, EXE, SHR, WRT, LONG
                                0000     .4
                            00000014    0000     .5 EXE$C_CMSTKSZ   = ^X14          ; Utterly unused Change mode call size
-3                                      0000      4 ;
                                        0000      5 ;*****************************************************************************
                                        0000      6 ;*                                                                           *
                                        0000     .1 ; COPYRIGHT (C) 1977, 1980
-1                                      0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                                        0000      9 ;*                                                                           *
                                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                                        0000     15 ;*  TRANSFERRED.                                                            *
                                        0000     16 ;*                                                                           *
                                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                                        0000     19 ;*  CORPORATION.                                                            *
                                        0000     20 ;*                                                                           *
                                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
                                        0000     23 ;*                                                                           *
                                        0000     24 ;*****************************************************************************
                                        0000     25 ;
                                        0000     26 ; D. N. CUTLER 15-SEP-76
                                        0000     27 ;
                                        0000     28 ; MODIFIED BY:
                                        0000     29 ;
                                        0000     .1 ;       DS5.4   John Ciukaj     5-May-1980      Version 5.4
                                        0000     .2 ;                 Used Vax/Vms version 2.0 SYSQIOFDT.MAR and created
                                        0000     .3 ;                 a UPD file with the edits necessary to reflect the
                                        0000     .4 ;                 changes exhibited in Supervisor version 5.3 module
                                        0000     .5 ;                 QIOFDT.
                                        0000     .6 ;
                                        0000     .7 ;                 Dave Butenhof   30-jun-1980,    version 5.5
                                        0000     .8 ;       02          Alter twc BSBW's to longword displacment JSB's since
                                        0000     .9 ;                 the supervisor's gotten fatter.
                                        0000    .10 ;
                                        0000    .11 ;       03        M. Baggett      8-Dec-1982      Version 6.10
                                        0000    .12 ;                 Change BSBW to JSB to fixed truncation error.
                                        0000    .13 ;
                                        0000    .14 ;
                                        0000     30 ;       SRB0003           Steve Beckhardt         23-Oct-1979
                                        0000     31 ;       Fixed bug to deallocate diagnostic buffer if QIO is being
                                        0000     32 ;       backed out to fault a page in.
                                        0000     33 ;
                                        0000     34 ;       MHB0029           M. H. Bramhall          28-Sep-1979
                                        0000     35 ;       Added EXE$CARRIAGE for carriage control processing.
                                        0000     36 ;
                                        0000     37 ;       SRB0002           STEVE BECKHARDT         30-MAR-1979
                                        0000     38 ;       ADDED ENTRY POINTS EXE$MODIFY, EXE$MODIFYLOCK, AND EXE$MODIFYLOCKR
                                        0000     39 ;       FOR FUNCTIONS THAT READ AND WRITE MEMORY.
                                        0000     40 ;
                                        0000     41 ;       SRB0001           STEVE BECKHARDT         29-MAR-1979
```

```
                                0000     42 ;       ADDED ROUTINES EXE$READLOCKR, EXE$WRITELOCKR, EXE$READCHKR, AND
                                0000     43 ;       EXE$WRITECHKR
                                0000     44 ;
                                0000     45 ; SYSTEM SERVICE QUEUE I/O FUNCTION DECISION TABLE SUBROUTINES
                                0000     46 ;
                                0000     47 ; MACRO LIBRARY CALLS
                                0000     48 ;
                                0000     49
                                0000     50         $ACBDEF                         ;DFFINE ACB OFFSETS
                                0000     51         $CCBDEF                         ;DEFINE CCB OFFSETS
                                0000     52         $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
                                0000     53         $IODEF                          ;DEFINE I/O FUNCTION CODES
                                0000     54         $IRPDEF                         ;DEFINE IRP OFFSETS
                                0000     55         $PCBDEF                         ;DEFINE PCB VALUES
                                C000     56         $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                                0000     57         $UCBDEF                         ;DEFINE UCB OFFSETS
                                0000     58         $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
                                0000     59
                                0000     60 ;
                                0000     61 ; LOCAL SYMBOLS
                                0000     62 ;
                                0000     63 ; ARGUMENT LIST OFFSET DEFINITIONS
                                0000     64 ;
                                0000     65
                    00000000    0000     66 P1=0                                    ;FIRST FUNCTION DEPENDENT PARAMETER
                    00000004    0000     67 P2=4                                    ;SECOND FUNCTION DEPENDENT PARAMETER
                    00000008    0000     68 P3=8                                    ;THIRD FUNCTION DEPENDENT PARAMETER
                    0000000C    0000     69 P4=12                                   ;FOURTH FUNCTION DEPENDENT PARAMETER
                    00000010    0000     70 P5=16                                   ;FIFTH FUNCTION DEPENDENT PARAMETER
                    00000014    0000     71 P6=20                                   ;SIXTH FUNCTION DEPENDENT PARAMETER
```

```
                                        0000     .2          .SBTTL  ONE PARAMETER FUNCTION PROCESSING
                                        0000     .3 ;+
                                        0000     .4 ; EXE$ONEPARM - ONE PARAMETER FUNCTION PROCESSING
                                        0000     .5 ;
                                        0000     .6 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
                                        0000     .7 ; PROCESS A ONE PARAMETER FUNCTION THAT REQUIRES NO SPECIAL CHECKING.
-16                                     0000     89 ;
                                        0000     90 ; INPUTS:
                                        0000     91 ;
                                        0000     .1 ;       R0 = SCRATCH.
                                        0000     .2 ;       R1 = SCRATCH.
                                        0000     .3 ;       R2 = SCRATCH.
                                        0000     .4 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                        0000     .5 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                        C000     .6 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                        0000     .7 ;       R6 = ADDRESS OF CCB.
                                        0000     .8 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                        0000     .9 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                        0000    .10 ;       R9 = SCRATCH.
                                        0000    .11 ;       R10 = SCRATCH.
                                        0000    .12 ;       R11 = SCRATCH.
                                        0000    .13 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
-5                                      0000     97 ;
                                        0000     98 ; OUTPUTS:
                                        0000     99 ;
                                        0000     .1 ; ***TBS***
                                        0000     .2 ;-
                                        0000     .3
                                        0000     .4          .ENABL  LSB
                                        0000     .5 EXE$ONEPARM::                                    ;ONE PARAMETER FUNCTION PROCESSING
 34 A3    6C    D0                     0000     .6          MOVL    P1(AP),IRP$L_MEDIA(R3)          ;STORE PARAMETER IN MEDIA ADDRESS
          03    11                     0004     .7          BRB     10$                             ;
```

```
                                0006     .9          .SBTTL  ZERO PARAMETER FUNCTION PROCESSING
                                0006    .10 ;+
                                0006    .11 ; EXE$ZEROPARAM - ZERO PARAMETER FUNCTION PROCESSING
                                0006    .12 ;
                                0006    .13 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
                                0006    .14 ; PROCESS A ZERO PARAMETER FUNCTION THAT REQUIRES NO ADDITION CHECKING.
-37                             0006    137 ;
                                0006    138 ; INPUTS:
                                0006    139 ;
                                0006    140 ;       R0 = SCRATCH.
                                0006    141 ;       R1 = SCRATCH.
                                0006    142 ;       R2 = SCRATCH.
                                0006    143 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                0006    144 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                C006    145 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                0006    146 ;       R6 = ADDRESS OF CCB.
                                0006    147 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                0006    148 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                0006    149 ;       R9 = SCRATCH.
                                0006    150 ;       R10 = SCRATCH.
                                0006    151 ;       R11 = SCRATCH.
                                0006    152 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                0006    153 ;
                                0006    154 ; OUTPUTS:
                                0006    155 ;
                                0006    156 ; ***TBS***
                                0006    157 ;-
                                0006    158
                                0006     .1 EXE$ZEROPARM::                                  ;ZERO PARAMETER FUNCTION PROCESSING
 34 A3   D4                     0006     .2          CLRL    IRP$L_MEDIA(R3)                ;CLEAR PARAMETER
  FFF4'  31                     0009     .3 10$:     BRW     EXE$QIODRVPKT                  ;QUEUE I/O PACKET TO DRIVER
                                000C     .4          .DSABL  LSB
```

```
                                000C     .6          .SBTTL  READ AND WRITE FUNCTION PROCESSING
                                000C     .7 ;+
                                000C     .8 ; EXE$READ - READ FUNCTION PROCESSING
                                000C     .9 ; EXE$WRITE - WRITE FUNCTION PROCESSING
                                000C    .10 ; EXE$MODIFY - MODIFY FUNCTION PROCESSING
                                000C    .11 ;
                                000C    .12 ; THESE ROUTINES ARE CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
                                000C    .13 ; PROCESS A READ OR WRITE PHYSICAL OR LOGICAL FUNCTION.
                                000C    .14 ; EXE$MODIFY IS USED FOR FUNCTIONS THAT READ AND WRITE MEMORY.
-11                             000C    170 ;
                                000C    171 ; INPUTS:
                                000C    172 ;
                                000C    173 ;       R0 = SCRATCH.
                                000C    174 ;       R1 = SCRATCH.
                                C00C    175 ;       R2 = SCRATCH.
                                000C    176 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                000C    177 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                000C    178 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                000C    179 ;       R6 = ADDRESS OF CCB.
                                000C    180 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                000C    181 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                000C    182 ;       R9 = SCRATCH.
                                000C    183 ;       R10 = SCRATCH.
                                000C    184 ;       R11 = SCRATCH.
                                000C    185 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                000C    186 ;
                                000C    187 ; OUTPUTS:
                                000C    188 ;
                                000C    189 ; ***TBS***
                                000C    190 ;-
                                000C    191
                                000C     .1          .ENABL  LSB
                                000C     .2 EXE$MODIFY::                                 ;MODIFY FUNCTION PROCESSING
    52    44'AF    DE           000C     .3          MOVAL   B^EXE$MODIFYLOCK,R2        ;SET ADDRESS OF BUFFER CHECK ROUTINE
          04    11               0010     .4          BRB     5$
                                 0012     .5 EXE$READ::                                   ;READ FUNCTION PROCESSING
    52    3E'AF    DE           0012     .6          MOVAL   B^EXE$READLOCK,R2          ;SET ADDRESS OF BUFFER CHECK ROUTINE
04  2A A3    01    E3           0016     .7 5$:      BBCS    #IRP$V_FUNC,IRP$W_STS(R3),10$ ;SET READ FUNCTION STATUS
                                 0018     .8 EXE$WRITE::                                  ;WRITE FUNCTION PROCESSING
    52    41'AF    DE           0018     .9          MOVAL   B^EXE$WRITELOCK,R2         ;SET ADDRESS OF BUFFER CHECK ROUTINE
38  A3    0C AC    D0           001F    .10 10$:     MOVL    P4(AP),IRP$B_CARCON(R3)    ;INSERT CARRIAGE CONTROL BYTE
      06    00    ED           0024    .11          CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;PHYSICAL I/O FUNCTION?
    1F    20 A3                0027    .12                  IRP$W_FUNC(R3),#IO$_PHYSICAL ;
          04    15               002A    .13          BLEQ    20$                        ;IF LEQ YES
          15    A2               002C    .14          SUBW    #IO$_READLBLK-IO$_READPBLK,- ;CONVERT TO PHYSICAL FUNCTION
          20 A3                  002E    .15                  IRP$W_FUNC(R3)             ;
    51    04 AC    3C           0030    .16 20$:     MOVZWL  P2(AP),R1                  ;GET NUMBER OF BYTES TO TRANSFER
          05    13               0034    .17          BEQL    30$                        ;IF EQL NONE
       50    6C    D0           0036    .18          MOVL    P1(AP),R0                  ;GET STARTING VIRTUAL ADDRESS OF TRANSFER
          62    16               0039    .19          JSB     (R2)                       ;CHECK BUFFER AND LOCK IN MEMORY
             FFC2'   31           003B    .20 30$:     BRW     EXE$QIODRVPKT              ;QUEUE I/O PACKET TO DRIVER
                                 003E    .21          .DSABL  LSB
```

```
                                003E    .23         .SBTTL  READ AND WRITE FUNCTION BUFFER CHECK AND LOCK ROUTINES
                                003E    .24 ;+
                                003E    .25 ; EXE$READLOCK - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK
                                003E    .26 ; EXE$WRITELOCK - CHECK BUFFER FOR WRITE ACCESSIBILITY AND LOCK
                                003E    .27 ; EXE$MODIFYLOCK - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK
                                003E    .28 ;
                                003E    .29 ; THESE ROUTINES ARE CALLED TO CHECK THE ACCESSIBILITY OF AN I/O BUFFER AND
                                003E    .30 ; TO LOCK THE BUFFER IN MEMORY FOR A DIRECT MEMORY TRANSFER.
-14                             003E    206 ;
                                003E    207 ; INPUTS:
                                003E    208 ;
                                003E     .1 ;       R0 = STARTING ADDRESS OF I/O BUFFER.
                                003E     .2 ;       R1 = LENGTH OF TRANSFER IN BYTES.
                                003E     .3 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                C03E     .4 ;       R6 = ADDRESS OF CCB.
-13                             003E    222 ;
                                003E    223 ; OUTPUTS:
                                003E    224 ;
-43                             003E    268 ;       THE I/O BUFFER IS CHECKED FOR THE PROPER ACCESSIBILITY. IF THE
                                003E    269 ;       CHECK SUCCEEDS, THEN THE BUFFER IS LOCKED IN MEMORY AND THE STARTING
                                003E    270 ;       ADDRESS OF THE PAGE TABLE ENTRIES THAT MAP THE TRANSFER IS STORED
                                003E    271 ;       IN THE I/O PACKET. ELSE THE I/O IS COMPLETED WITH A STATUS OF
                                003E    272 ;       ACCESS VIOLATION.
                                003E    273 ;-
                                003E    274
                                003E    275 EXE$READLOCK::                              ;CHECK BUFFER FOR READ FUNCTION AND LOCK
                     11   10    003E    276         BSBB    EXE$READLOCKR               ;EXE$READLOCKR RETURNS NORMALLY ON
                                0040    277                                             ;SUCCESS, VIA COROUTINE CALL ON FAILURE
                          05    0040    278         RSB                                 ;RETURNS TO CALLER ON SUCCESS, TO
                                0041    279                                             ;EXE$READLOCKR ON FAILURE
                                0041    280
                                0041    281 EXE$WRITELOCK::                             ;CHECK BUFFER FOR WRITE FUNCTION AND LOCK
                     15   10    0041    282         BSBB    EXE$WRITELOCKR              ;EXE$WRITELOCKR RETURNS NORMALLY ON
                                0043    283                                             ;SUCCESS, VIA COROUTINE CALL ON FAILURE
                          05    0043    284         RSB                                 ;RETURNS TO CALLER ON SUCCESS, TO
                                0044    285                                             ;EXE$WRITELOCKR ON FAILURE
                                0044    286
                                0044    287 EXE$MODIFYLOCK::                            ;CHECK BUFFER FOR MODIFY FUNCTION AND LOCK
                     01   10    0044    288         BSBB    EXE$MODIFYLOCKR             ;EXE$MODIFYLOCKR RETURNS NORMALLY ON
                                0046    289                                             ;SUCCESS, VIA COROUTINE CALL ON FAILURE
                          05    0046    290         RSB                                 ;RETURNS TO CALLER ON SUCCESS, TO
                                0047    291                                             ;EXE$MODIFYLOCKR ON FAILURE
```

```
                                0047   293          .SBTTL  READ AND WRITE BUFFER CHECK AND LOCK AND RETURN ROUTINES
                                0047   294 ;+
                                0047   295 ; EXE$READLOCKR - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK AND RETURN
                                0047   296 ;                 ON ERROR
                                0047   297 ; EXE$WRITELOCKR - CHECK BUFFER FOR WRITE ACCESSIBILITY AND LOCK AND RETURN
                                0047   298 ;                  ON ERROR
                                0047   299 ; EXE$MODIFYLOCKR - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK AND RETURN
                                0047   300 ;                   ON ERROR
                                0047   301 ;
                                0047   302 ; THESE ROUTINES ARE CALLED TO CHECK THE ACCESSIBILITY OF AN I/O BUFFER
                                0047   303 ; AND TO LOCK THE BUFFER IN MEMORY FOR A DIRECT MEMORY TRANSFER.  IN
                                0047   304 ; ADDITION, THESE ROUTINES PERFORM A COROUTINE CALL IF THERE IS AN ERROR
                                0047   305 ; OR ANY PAGES HAVE TO BE FAULTED IN.  THE PURPOSE OF THE COROUTINE
                                0047   306 ; CALL IS TO ALLOW THE CALLER TO PERFORM ANY NECESSARY CLEANUP BEFORE
                                0047   307 ; THE QIO IS BACKED UP OR ABORTED.  THESE ROUTINES ARE TYPICALLY CALLED
                                0047   308 ; BY DRIVERS THAT MUST LOCK MULTIPLE AREAS INTO MEMORY.  SINCE THESE
                                0047   309 ; ROUTINES CANNOT UNLOCK AREAS PREVIOUSLY LOCKED, THE COROUTINE CALL ALLOWS
                                0047   310 ; THE CALLER (THE DRIVER) TO UNLOCK PREVIOUSLY LOCKED AREAS (AND PERFORM
                                0047   311 ; ANY OTHER CLEANUP) AND THEN RETURN HERE TO BACK UP OR ABORT THE I/O.
                                0047   312 ;
                                0047   313 ; EXE$MODIFYLOCKR IS USED WHEN THE BUFFER WILL BE READ AND WRITTEN BY THE
                                0047   314 ; I/O DEVICE.  IT DISABLES AN OPTIMIZATION IN MMG$IOLOCK WHICH IS USED
                                0047   315 ; WHEN THE BUFFER IS ONLY WRITTEN.
                                0047   316 ;
                                0047   317 ; INPUTS:
                                0047   318 ;
                                0047   319 ;       R0 = STARTING ADDRESS OF I/O BUFFER.
                                0047   320 ;       R1 = LENGTH OF BUFFER IN BYTES.
                                0047   321 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                0047   322 ;       R6 = ADDRESS OF CCB.
                                0047   323 ;
                                0047   324 ; OUTPUTS:
                                0047   325 ;
                                0047   326 ;       THE I/O BUFFER IS CHECKED FOR THE PROPER ACCESSIBILITY.  IF THE
                                0047   327 ;       CHECK SUCCEEDS, THEN THE BUFFER IS LOCKED IN MEMORY AND THE STARTING
                                0047   328 ;       ADDRESS OF THE PAGE TABLE ENTRIES THAT MAP THE TRANSFER IS STORED
                                0047   329 ;       IN THE I/O PACKET.
                                0047   330 ;
                                0047   331 ;       R0 = RETURN CODE
                                0047   332 ;
                                0047   333 ;       NOTE THAT IF THERE ARE NO ERRORS AND NO PAGES HAVE TO BE FAULTED
                                0047   334 ;       IN, THEN THESE ROUTINES RETURN NORMALLY.  HOWEVER, IF THERE IS AN
                                0047   335 ;       ERROR OR A PAGE HAS TO BE FAULTED IN, THEN THE CALLER IS CALLED
                                0047   336 ;       BY A COROUTINE CALL.  THE CALLER'S RSB THEN RETURNS HERE WHERE
                                0047   337 ;       THE QIO IS EITHER BACKED UP OR ABORTED.  NOTE THAT IN THIS CASE
                                0047   338 ;       THE CALLER'S ERROR HANDLING CODE MUST PRESERVE ALL REGISTERS,
                                0047   339 ;       INCLUDING R0 AND R1.
                                0047   340 ;-
                                0047   341          .ENABL  LSB
                                0047   342 EXE$MODIFYLOCKR::                           ;CHECK BUFFER FOR MODIFY FUNCTION AND LOCK
     50    DD                   0047   343          PUSHL   R0                         ;SAVE STARTING ADDRESS OF BUFFER
   0096    30                   0049   344          BSBW    EXE$READCHKR               ;CHECK BUFFER FOR READ FUNCTION
52   04    C8                   004C   345          BISL    #4,R2                      ;DISABLE OPTIMIZATION IN MMG$IOLOCK
     0C    11                   004F   346          BRB     10$
                                0051   347
                                0051   348 EXE$READLOCKR::                            ;CHECK BUFFER FOR READ FUNCTION AND LOCK
     50    DD                   0051   349          PUSHL   R0                         ;SAVE STARTING ADDRESS OF BUFFER
```

```
                 008C   30  0053   350          BSBW    EXE$READCHKR                ;CHECK BUFFER FOR READ FUNCTION
                   05   11  0056   351          BRB     10$
                        0058   352
                        0058   353 EXE$WRITELOCKR::                                ;CHECK BUFFER FOR WRITE FUNCTION AND LOCK
                   50   DD  0058   354          PUSHL   R0                          ;SAVE STARTING ADDRESS OF BUFFER
                 00AF   30  005A   355          BSBW    EXE$WRITECHKR               ;CHECK BUFFER FOR WRITE FUNCTION
               1D 50    E9  005D   356 10$:     BLBC    R0,15$                      ;BRANCH IF ERROR
                  50 8ED0  0060   357          POPL    R0                          ;RESTORE STARTING ADDRESS OF BUFFER
          50   FE00 8F    AB  0063   358          BICW3   #^C<VA$M_BYTE>,R0,IRP$W_BOFF(R3) ;SET BYTE OFFSET IN PAGE
               30 A3            0068
                   53   DD  006A   359          PUSHL   R3                          ;SAVE ADDRESS OF I/O PACKET
        00000000'EF     16  006C    .1          JSB     MMG$IOLOCK                  ;LOCK PAGES FOR I/O                         [03]
-1           53   8E   D0  0072   361          MOVL    (SP)+,R3                    ;RETRIEVE ADDRESS OF I/O PACKET
                08 50    E9  0075   362          BLBC    R0,20$                      ;IF LBC LOCK FAILURE
             2C A3   51   D0  0078   363          MOVL    R1,IRP$L_SVAPTE(R3)         ;INSERT ADDRESS OF FIRST PTE IN PACKET
                         05  007C   364          RSB                                 ;
                5E   04   C0  007D   365 15$:    ADDL    #4,SP                       ;THROW AWAY OLD R0
                     9E   16  0080   366 20$:    JSB     @(SP)+                      ;COROUTINE CALL TO CLEANUP
                     50   D5  0082   367          TSTL    R0                          ;ERRORS ENCOUNTERED?
                     45   12  0084   368          BNEQ    50$                         ;IF NEQ YES
                     51   DD  0086   369          PUSHL   R1                          ;SAVE VIRTUAL ADDRESS OF PAGE TO FAULT
                   3E A4  B6  0088   370          INCW    PCB$W_DIOCNT(R4)            ;ADJUST DIRECT I/O COUNT
                   0A A6  B7  008B   371          DECW    CCB$W_IOC(R6)               ;DECREMENT CHANNEL I/O COUNT
          0F 2A A3   07   E1  008E   372          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),25$  ;BR. IF NO DIAGNOSTIC BUFFER
             50   44 A3   D0  0093   373          MOVL    IRP$L_DIAGBUF(R3),R0        ;GET ADDRESS OF DIAGNOSTIC BUFFER
                     53   DD  0097   374          PUSHL   R3                          ;SAVE R3
        00000000'EF     16  0099    .1          jsb     l^exe$deanonpaged                   ; Deallocate diagnostic buffer
-1                   53 8ED0  009F   376          POPL    R3                          ;RESTORE R3
                            00A2   377 25$:
          03 0B A3   06   E1  00A2   378          BBC     #ACB$V_QUOTA,IRP$B_RMOD(R3),30$ ;IF CLR, NO AST SPECIFIED
                  38 A4   B6  00A7   379          INCW    PCB$W_ASTCNT(R4)            ;ADJUST AST COUNT
                50   53   D0  00AA   380 30$:     MOVL    R3,R0                       ;SET ADDRESS OF BLOCK TO DEALLOCATE
        00000000'EF     16  00AD    .1          jsb     l^exe$deanonpaged                   ; Deallocate I/O packet
-1                   02   BA  00B3   382          POPR    #^M<R1>                     ;RETRIEVE VIRTUAL ADDRESS OF PAGE TO FAULT
                5E   5D   D0  00B5   383          MOVL    FP,SP                       ;TRIM STACK BACK TO CHANGE MODE FRAME
             5C   08 AD   7D  00B8   384          MOVQ    8(FP),AP                    ;RESTORE USER ARGUMENT AND FRAME POINTERS
                5E   14   C0  00BC   385          ADDL    S^#EXE$C_CMSTKSZ,SP         ;REMOVE CHANGE MODE CALL FRAME FROM STACK
          50    8E   04   C3  00BF   386          SUBL3   #4,(SP)+,R0                 ;CALCULATE RESTART ADDRESS
                  C7'AF   9F  00C3   387          PUSHAB  B^40$                       ;SET NEW RETURN ADDRESS
                          02  00C6   388          REI                                 ;
                     61   95  00C7   389 40$:     TSTB    (R1)                        ;FAULT USER BUFFER AGAIN
                     60   17  00C9   390          JMP     (R0)                        ;REPEAT SYSTEM SERVICE
                   FF32'  31  00CB   391 50$:     BRW     EXE$ABORTIO                 ;ABORT I/O REQUEST
                          00CE   392          .DSABL  LSB
```

```
                        00CE    394          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
                        00CE    395 ;+
                        00CE    396 ; EXE$READCHK - CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
                        00CE    397 ;
                        00CE    398 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A READ I/O
                        00CE    399 ; FUNCTION.
                        00CE    400 ;
                        00CE    401 ; INPUTS:
                        00CE    402 ;
                        00CE    403 ;       R0 = ADDRESS OF BUFFER.
                        00CE    404 ;       R1 = SIZE OF TRANSFER IN BYTES.
                        00CE    405 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        00CE    406 ;
                        00CE    407 ; OUTPUTS:
                        00CE    408 ;
                        00CE    409 ;       IF BUFFER IS NOT WRITE ACCESSIBLE, THEN THE I/O REQUEST IS TERM-
                        00CE    410 ;       INATED VIA EXE$IOFINISH WITH A STATUS OF SS$_ACCVIO.
                        00CE    411 ;
                        00CE    412 ;       IF BUFFER IS WRITE ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                        00CE    413 ;       TURNED:
                        00CE    414 ;
                        00CE    415 ;               R0 = ADDRESS OF BUFFER.
                        00CE    416 ;               R1 = SIZE OF TRANSFER IN BYTES.
                        00CE    417 ;               R2 = READ FUNCTION INDICATOR (1).
                        00CE    418 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                        00CE    419 ;
                        00CE    420 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                        00CE    421 ;               IRP$W_FUNC(R3) = READ.
                        00CE    422 ;-
                        00CE    423
                        00CE    424          .ENABL  LSB
                        00CE    425 EXE$READCHK::                               ;CHECK BUFFER FOR READ FUNCTION
50      DD              00CE    426          PUSHL   R0                         ;SAVE ADDRESS OF BUFFER
10      10              00D0    427          BSBB    EXE$READCHKR               ;CHECK BUFFER
04      11              00D2    428          BRB     10$
```

```
                                00D4    430          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
                                00D4    431 ;+
                                00D4    432 ; EXE$WRITECHK - CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
                                00D4    433 ;
                                00D4    434 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A WRITE I/O
                                00D4    435 ; FUNCTION.
                                00D4    436 ;
                                00D4    437 ; INPUTS:
                                00D4    438 ;
                                00D4    439 ;       R0 = ADDRESS OF BUFFER.
                                00D4    440 ;       R1 = SIZE OF TRANSFER IN BYTES.
                                00D4    441 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                00D4    442 ;
                                00D4    443 ; OUTPUTS:
                                C0D4    444 ;
                                00D4    445 ;       IF BUFFER IS NOT READ ACCESSIBLE, THEN THE I/O REQUEST IS TERM-
                                00D4    446 ;       INATED VIA EXE$IOFINISH WITH A STATUS OF SS$_ACCVIO.
                                00D4    447 ;
                                00D4    448 ;       IF BUFFER IS READ ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                                00D4    449 ;       TURNED:
                                00D4    450 ;
                                00D4    451 ;               R0 = ADDRESS OF BUFFER.
                                00D4    452 ;               R1 = SIZE OF TRANSFER IN BYTES.
                                00D4    453 ;               R2 = WRITE FUNCTION INDICATOR (0).
                                00D4    454 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                                00D4    455 ;
                                00D4    456 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                                00D4    457 ;               IRP$W_FUNC(R3) = WRITE.
                                00D4    458 ;-
                                00D4    459
                                00D4    460 EXE$WRITECHK::                              ;CHECK BUFFER FOR WRITE FUNCTION
    50    DD                    00D4    461          PUSHL   R0                         ;SAVE ADDRESS OF BUFFER
    34    10                    00D6    462          BSBB    EXE$WRITECHKR              ;CHECK BUFFER
03 50     E8                    00D8    463 10$:     BLBS    R0,20$                     ;BRANCH IF SUCCESS
  FF22'   31                    00DB    464          BRW     EXE$ABORTIO                ;ABORT I/O
    50 8ED0                     00DE    465 20$:     POPL    R0                         ;RESTORE ADDRESS OF BUFFER
       05                       00E1    466          RSB
                                00E2    467          .DSABL  LSB
```

```
                                00E2    469          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
                                00E2    470 ;+
                                00E2    471 ; EXE$READCHKR - CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
                                00E2    472 ;
                                00E2    473 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A READ I/O
                                00E2    474 ; FUNCTION.  STATUS IS RETURNED IN R0.
                                00E2    475 ;
                                00E2    476 ; INPUTS:
                                00E2    477 ;
                                00E2    478 ;       R0 = ADDRESS OF BUFFER.
                                00E2    479 ;       R1 = SIZE OF TRANSFER IN BYTES.
                                00E2    480 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                00E2    481 ;
                                00E2    482 ; OUTPUTS:
                                00E2    483 ;
                                00E2    484 ;       IF THE BUFFER IS NOT WRITE ACCESSIBLE, THEN THE FOLLOWING
                                00E2    485 ;       VALUE IS RETURNED:
                                00E2    486 ;
                                00E2    487 ;       R0 = SS$_ACCVIO
                                00E2    488 ;
                                00E2    489 ;       IF BUFFER IS WRITE ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                                00E2    490 ;       TURNED:
                                00E2    491 ;
                                00E2    492 ;               R0 = SS$_NORMAL
                                00E2    493 ;               R1 = SIZE OF TRANSFER IN BYTES.
                                00E2    494 ;               R2 = READ FUNCTION INDICATOR (1).
                                00E2    495 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                                00E2    496 ;
                                00E2    497 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                                00E2    498 ;               IRP$W_FUNC(R3) = READ.
                                00E2    499 ;-
                                00E2    500
                                00E2    501          .ENABL  LSB
                                00E2    502 EXE$READCHKR::                              ;CHECK BUFFER FOR READ FUNCTION
          51    DD              00E2    503          PUSHL   R1                         ;SAVE R1
     51   50    C0              00E4    504          ADDL    R0,R1                      ;CALCULATE ENDING BUFFER ADDRESS
50   01FF 8F    AA              00E7    505          BICW    #VA$M_BYTE,R0              ;TRUNCATE TO START OF PAGE
     51   50    C2              00EC    506          SUBL    R0,R1                      ;CALCULATE LENGTH OF BUFFER TO PROBE
52   FE00 8F    32              00EF    507          CVTWL   #-^X200,R2                 ;SET ADDRESS ADJUSTMENT CONSTANT
                                00F4    508 10$:     IFNOWRT R1,(R0),ACCESS             ;CAN ENDS OF BUFFER BE WRITTEN?
     50   52    C2              00FA    509          SUBL    R2,R0                      ;CALCULATE VIRTUAL ADDRESS OF NEXT PAGE
   51   6142    3E              00FD    510          MOVAW   (R1)[R2],R1                ;CALCULATE NEW LENGTH
          F1    14              0101    511          BGTR    10$                        ;IF GTR MORE TO TEST
 2A A3    02    A8              0103    512          BISW    #IRP$M_FUNC,IRP$W_STS(R3)  ;SET READ FUNCTION
     52   01    D0              0107    513          MOVL    #1,R2                      ;SET READ FUNCTION INDICATOR
          23    11              010A    514          BRB     30$                        ;
```

```
                                010C   516          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
                                010C   517 ;+
                                010C   518 ; EXE$WRITECHKR - CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
                                010C   519 ;
                                010C   520 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A WRITE I/O
                                010C   521 ; FUNCTION.  STATUS IS RETURNED IN R0
                                010C   522 ;
                                010C   523 ; INPUTS:
                                010C   524 ;
                                010C   525 ;       R0 = ADDRESS OF BUFFER.
                                010C   526 ;       R1 = SIZE OF TRANSFER IN BYTES.
                                010C   527 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                010C   528 ;
                                010C   529 ; OUTPUTS:
                                C10C   530 ;
                                010C   531 ;       IF BUFFER IS NOT READ ACCESSIBLE, THEN THE FOLLOWING VALUE IS
                                010C   532 ;       RETURNED:
                                010C   533 ;
                                010C   534 ;       R0 = SS$_ACCVIO
                                010C   535 ;
                                010C   536 ;       IF BUFFER IS READ ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                                010C   537 ;       TURNED:
                                010C   538 ;
                                010C   539 ;               R0 = SS$_NORMAL
                                010C   540 ;               R1 = SIZE OF TRANSFER IN BYTES.
                                010C   541 ;               R2 = WRITE FUNCTION INDICATOR (0).
                                010C   542 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                                010C   543 ;
                                010C   544 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                                010C   545 ;               IRP$W_FUNC(R3) = WRITE.
                                010C   546 ;-
                                010C   547
                                010C   548 EXE$WRITECHKR::                              ;CHECK BUFFER FOR WRITE FUNCTION
                      51    DD  010C   549          PUSHL   R1                        ;SAVE R1
                51    50    C0  010E   550          ADDL    R0,R1                     ;CALCULATE ENDING BUFFER ADDRESS
           50   01FF   8F    AA  0111   551          BICW    #VA$M_BYTE,R0             ;TRUNCATE TO START OF PAGE
                51    50    C2  0116   552          SUBL    R0,R1                     ;CALCULATE LENGTH OF BUFFER TO PROBE
           52   FE00   8F    32  0119   553          CVTWL   #-^X200,R2                ;GET ADDRESS ADJUSTMENT CONSTANT
                                011E   554 20$:      IFNORD  R1,(R0),ACCESS            ;CAN ENDS OF BUFFER BE READ?
                50    52    C2  0124   555          SUBL    R2,R0                     ;CALCULATE VIRTUAL ADDRESS OF NEXT PAGE
                51  6142    3E  0127   556          MOVAW   (R1)[R2],R1               ;CALCULATE NEW LENGTH
                      F1    14  012B   557          BGTR    20$                       ;IF GTR MORE TO CHECK
                      52    D4  012D   558          CLRL    R2                        ;SET WRITE FUNCTION INDICATOR
                      51  8ED0  012F   559 30$:      POPL    R1                        ;RESTORE R1
             32   A3    51    B0  0132   560          MOVW    R1,IRP$W_BCNT(R3)         ;INSERT TRANSFER BYTE COUNT IN PACKET
                50    01    3C  0136   561          MOVZWL  #SS$_NORMAL,R0            ;SUCCESS RETURN
                            05  0139   562          RSB                               ;
                                013A   563 ACCESS:                                    ;SET ACCESS VIOLATION
                      51  8ED0  013A   564          POPL    R1                        ;RESTORE R1
                50    0C    3C  013D   565          MOVZWL  #SS$_ACCVIO,R0            ;
                            05  0140   566          RSB
                                0141   567          .DSABL  LSB
```

```
                                0141    569          .SBTTL  SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
                                0141    570 ;+
                                0141    571 ; EXE$SETCHAR - SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
                                0141    572 ;
                                0141    573 ; THIS ROUTINE PLACES THE NEW CHARACTERISTICS SPECIFIED BY THE QUADWORD POINTED
                                0141    574 ; TO BY P1 INTO THE SECOND AND THIRD LONGWORDS OF THE DEVICE UCB.
                                0141    575 ;
                                0141    576 ; INPUTS:
                                0141    577 ;
                                0141    578 ;       R0 = SCRATCH.
                                0141    579 ;       R1 = SCRATCH.
                                0141    580 ;       R2 = SCRATCH.
                                0141    581 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                0141    582 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                0141    583 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                0141    584 ;       R6 = ADDRESS OF CCB.
                                0141    585 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                0141    586 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                0141    587 ;       R9 = SCRATCH.
                                0141    588 ;       R10 = SCRATCH.
                                0141    589 ;       R11 = SCRATCH.
                                0141    590 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                0141    591 ;
                                0141    592 ; OUTPUTS:
                                0141    593 ;
                                0141    594 ;       THE CHARACTERISTICS SPECIFIED BY THE QUADWORD POINTER TO BY P1 ARE STORED
                                0141    595 ;       IN THE SECOND AND THIRD LONGWORDS OF THE DEVICE UCB.
                                0141    596 ;-
                                0141    597
                                0141    598          .ENABL  LSB
                                0141    599 EXE$SETCHAR::                                   ;SET DEVICE MODE AND CHARACTERISTICS
        51    6C    D0          0141    600          MOVL    P1(AP),R1                      ;GET ADDRESS OF CHARACTERISTICS
                                0144    601          IFNORD  #8,(R1),ACCESS                 ;CAN CHARACTERISTICS QUADWORD BE READ?
        57    23    D1          014A    602          CMPL    #IO$_SETMODE,R7                ;SET MODE FUNCTION?
              04    13          014D    603          BEQL    10$                            ;IF EQL YES
     38 A5    61    B0          014F    604          MOVW    (R1),UCB$B_DEVCLASS(R5)        ;SET DEVICE TYPE AND CLASS
3A   A5    02 A1    B0          0153    605 10$:     MOVW    2(R1),UCB$W_DEVBUFSIZ(R5)      ;SET DEFAULT BUFFER SIZE
     A5    04 A1    D0          0158    606          MOVL    4(R1),UCB$L_DEVDEPEND(R5)      ;SET DEVICE CHARACTERISTICS
              14    11          015D    607          BRB     20$                            ;
```

```
                        015F   609          .SBTTL  SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS
                        015F   610 ;+
                        015F   611 ; EXE$SETMODE - SET DEVICE CHARACTERISTICS AND MODE
                        015F   612 ;
                        015F   613 ; FUNCTIONAL DESCRIPTION:
                        015F   614 ;
                        015F   615 ; THIS ROUTINE PLACES THE NEW CHARACTERISTICS SPECIFIED BY P1 INTO
                        015F   616 ; THE I/O PACKET FOR INSERTION INTO THE UCB WHEN THE UNIT IS IDLE.
                        015F   617 ; THE INPUT DATA IS IN THE FORM RETURNED BY $GTCHAN. THE SPECIFIED BUFFER
                        015F   618 ; IS ASSUMED TO BE 12 BYTES IN LENGTH. THE P2 LENGTH SPECIFIER IS IGNORED.
                        015F   619 ;
                        015F   620 ; THE NEW CHARACTERISTICS ARE PLACED IN IRP$L_MEDIA/MEDIA+4 AND THE
                        015F   621 ; PACKET IS QUEUED VIA EXE$QIODRVPKT.
                        015F   622 ;
                        015F   623 ; INPUTS:
                        015F   624 ;
                        015F   625 ;       R3 = I/O PACKET ADDRESS
                        015F   626 ;       R4 = CURRENT PCB
                        015F   627 ;       R5 = ACB ADDRESS
                        015F   628 ;       R6 = ASSIGNED CCB ADDRESS
                        015F   629 ;       AP = ADDRESS OF THE QIO ARGUMEMT P1
                        015F   630 ;
                        015F   631 ; OUTPUTS:
                        015F   632 ;
                        015F   633 ;       R0 = STATUS OF THE OPERATION
                        015F   634 ;       R3+ ARE PRESERVED.
                        015F   635 ;
                        015F   636 ; COMPLETION CODES:
                        015F   637 ;
                        015F   638 ;       SS$_NORMAL - SUCCESSFUL
                        015F   639 ;       SS$_ACCVIO - BUFFER ACCESS VIOLATION
                        015F   640 ;-
                        015F   641
                        015F   642 EXE$SETMODE::                                ;SET DEVICE MODE AND CHARACTERISTICS
    51   6C    D0       015F   643          MOVL    P1(AP),R1                  ;ADDRESS BUFFER
                        0162   644          IFNORD  #8,(R1),ACCESS             ;BR IF NO ACCESS
34 A3   61    7D       0168   645          MOVQ    (R1),IRP$L_MEDIA(R3)       ;INSERT CHARACTERISTICS IN I/O PACKET
     FE91'    31       016C   646          BRW     EXE$QIODRVPKT              ;QUEUE THE PACKET
```

```
                                016F    648          .SBTTL  SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
                                016F    649 ;+
                                016F    650 ; EXE$SENSEMODE - SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
                                016F    651 ;
                                016F    652 ; THIS ROUTINE OBTAINS THE CURRENT DEVICE MODE/CHARACTERISTICS FROM THE DEVICE
                                016F    653 ; DEPENDENT CHARACTERISTICS LONGWORD IN THE UCB AND IMMEDIATELY COMPLETES THE
                                016F    654 ; I/O OPERATION WITH THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE
                                016F    655 ; DEVICE DEPENDENT CHARACTERSITICS.
                                016F    656 ;
                                016F    657 ; INPUTS:
                                016F    658 ;
                                016F    659 ;       R0 = SCRATCH.
                                016F    660 ;       R1 = SCRATCH.
                                016F    661 ;       R2 = SCRATCH.
                                C16F    662 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                016F    663 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                016F    664 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                016F    665 ;       R6 = ADDRESS OF CCB.
                                016F    666 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                016F    667 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                016F    668 ;       R9 = SCRATCH.
                                016F    669 ;       R10 = SCRATCH.
                                016F    670 ;       R11 = SCRATCH.
                                016F    671 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                016F    672 ;
                                016F    673 ; OUTPUTS:
                                016F    674 ;
                                016F    675 ;       THE DEVICE DEPENDENT CHARACTERISTICS ARE OBTAINED FROM THE UCB AND
                                016F    676 ;       THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD EQUAL TO THE
                                016F    677 ;       DEVICE CHARACTERISTICS.
                                016F    678 ;-
                                016F    679
                                016F    680 EXE$SENSEMODE::                                 ;SENSE DEVICE MODE/CHARACTERISTICS
 51    3C A5    D0              016F    681          MOVL    UCB$L_DEVDEPEND(R5),R1         ;GET DEVICE DEPENDENT CHARACTERSITICS
    50    01    3C              0173    682 20$:     MOVZWL  #SS$_NORMAL,R0                 ;SET NORMAL COMPLETION STATUS
       FE87'    31              0176    683          BRW     EXE$FINISHIO                   ;FINISH I/O OPERATION
                                0179    684          .DSABL  LSB
```

```
                     0179   686          .SBTTL  CARRIAGE CONTROL INTERPRETATION
                     0179   687 ;+
                     0179   688 ; EXE$CARRIAGE - INTERPRET CARRIAGE CONTROL SPECIFIER
                     0179   689 ;
                     0179   690 ; FUNCTIONAL DESCRIPTION:
                     0179   691 ;
                     0179   692 ; THIS ROUTINE IS USED BY THE LINE PRINTER DRIVER AND THE TERMINAL
                     0179   693 ; DRIVER TO INTERPRET THE CARRIAGE CONTROL SPECIFIER IN IRP$B_CARCON .
                     0179   694 ; NOTE THAT IRP$B_CARCON IS USED AS A LONGWORD!
                     0179   695 ;
                     0179   696 ; THE SPECIFIER IS AS FOLLOWS:
                     0179   697 ;
                     0179   698 ;       .BYTE 1 -- FORTRAN CARRIAGE CONTROL CHARACTER IF NOT 0
                     0179   699 ;       .BYTE 2 -- ***** IGNORED *****
                     0179   700 ;       .BYTE 3 -- PREFIX CARRIAGE CONTROL
                     0179   701 ;       .BYTE 4 -- SUFFIX CARRIAGE CONTROL
                     0179   702 ;
                     0179   703 ; THE PRE/SUF FIELDS ARE AS FOLLOWS
                     0179   704 ;
                     0179   705 ;       IF BIT 7=0 THEN BITS 6-0 ARE THE NUMBER OF NEWLINES TO INSERT.
                     0179   706 ;       IF BIT 7=1 AND BIT 6=0 THEN BITS 4-0 ARE THE ASCII CHARACTER TO
                     0179   707 ;                                   OUTPUT. ASCII SET C0 OR C1 IS SPECIFIED BY BIT 5.
                     0179   708 ;       IF BIT 7=1 AND BIT 6=1 THEN BITS 5-0 ARE THE PRINTER CHANNEL NUMBER
                     0179   709 ;
                     0179   710 ; ASCII SET C0 IS ASSUMED AND BIT 6 IS IGNORED IF BIT 7=0.
                     0179   711 ;
                     0179   712 ; INPUTS:
                     0179   713 ;
                     0179   714 ;       R3 = ADDRESS OF THE I/O PACKET
                     0179   715 ;       R5 = ADDRESS OF THE UCB
                     0179   716 ;
                     0179   717 ; OUTPUTS:
                     0179   718 ;
                     0179   719 ;       IRP$B_CARCON IS SET UP TO REFLECT THE PRE/SUF CHARACTERS TO SEND.
                     0179   720 ;
                     0179   721 ;       BYTE 0 = NUMBER OF CHARACTERS TO SEND
                     0179   722 ;       BYTE 1 = CHARACTER, IF 0 THEN NEWLINE
                     0179   723 ;
                     0179   724 ;       IRP$B_CARCON+2 HAS THE SUFFIX CONTROL.
                     0179   725 ;
                     0179   726 ;       R0,R1 ARE USED.
                     0179   727 ;
                     0179   728 ;-
                     0179   729 ;
                     0179   730 ; LOCAL DATA TABLE
                     0179   731 ;
                     0179   732 CCTABLE:                             ; CARRIAGE CONTROL TO FORTRAN MATCH TABLE
 0D 01 00 01         0179   733          .BYTE   1,0,1,13            ; SPACE => 1 NL, 1 CR
          20         017D   734          .ASCII  / /
 0D 01 00 02         017E   735          .BYTE   2,0,1,13            ; "0" => 2 NL, 1 CR
          30         0182   736          .ASCII  /0/
 0D 01 0C 01         0183   737          .BYTE   1,12,1,13           ; "1" => 1 FF, 1 CR
          31         0187   738          .ASCII  /1/
 0D 01 00 00         0188   739          .BYTE   0,0,1,13            ; "+" => NOTHING, 1 CR
          2B         018C   740          .ASCII  /+/
 00 00 00 01         018D   741          .BYTE   1,0,0,0             ; "$" => 1 NL, NOTHING
          24         0191   742          .ASCII  /$/
```

```
                     0D 01 00 01  0192   743          .BYTE   1,0,1,13                ; DEFAULT => 1 NL, 1 CR
                        00        0196   744          .BYTE   0                       ; TABLE END
                                  0197   745 ;
                                  0197   746 ;
                                  0197   747 ;
                                  0197   748 EXE$CARRIAGE::                          ; INTERPRET CARRIAGE CONTROL
                  51    38 A3     9A     0197   749          MOVZBL  IRP$B_CARCON(R3),R1    ; GET FORTRAN SPECIFIER
                           12     13     019B   750          BEQL    20$                    ; IF EQL THEN TRY PRE/SUF
                  50    D9 AF     9E     019D   751          MOVAB   B^CCTABLE,R0           ; ADDRESS MATCH TABLE
                  38 A3    80     D0     01A1   752 10$:      MOVL    (R0)+,IRP$B_CARCON(R3) ; ASSUME MATCH
                           60     95     01A5   753          TSTB    (R0)                   ; END OF TABLE?
                           05     13     01A7   754          BEQL    15$                    ; IF EQL THEN YES
                     51    80     91     01A9   755          CMPB    (R0)+,R1               ; MATCH?
                           F3     12     01AC   756          BNEQ    10$                    ; NO THEN SEARCH
                                 05     01AE   757 15$:      RSB                            ; ELSE RETURN
                                        01AF   758 ;
                                        01AF   759 ; PRE/SUF CARRIAGE CONTROL
                                        01AF   760 ;
                  51    3A A3     9A     01AF   761 20$:      MOVZBL  IRP$B_CARCON+2(R3),R1  ; GET PREFIX SPECIFIER
                           02     13     01B3   762          BEQL    30$                    ; IF EQL THEN NONE
                           19     10     01B5   763          BSBB    100$                   ; INTERPRET THE SPECIFIER
                  38 A3    51     90     01B7   764 30$:      MOVB    R1,IRP$B_CARCON(R3)    ; INSERT NUMBER
                  39 A3    50     90     01BB   765          MOVB    R0,IRP$B_CARCON+1(R3)  ; INSERT CHARACTER
                  51    3B A3     9A     01BF   766          MOVZBL  IRP$B_CARCON+3(R3),R1  ; GET SUFFIX SPECIFIER
                           02     13     01C3   767          BEQL    40$                    ; IF EQL THEN NONE
                           09     10     01C5   768          BSBB    100$                   ; CONVERT THE SPECIFIER
                  3A A3    51     90     01C7   769 40$:      MOVB    R1,IRP$B_CARCON+2(R3)  ; INSERT NUMBER
                  3B A3    50     90     01CB   770          MOVB    R0,IRP$B_CARCON+3(R3)  ; INSERT CHARACTER
                                 05     01CF   771          RSB                            ; RETURN
                                        01D0   772 ;
                                        01D0   773 ; SUBROUTINE TO INTERPRET PRE/SUF SPECIFIER
                                        01D0   774 ;
                           50    D4     01D0   775 100$:     CLRL    R0                     ; ASSUME NEWLINE
                  08 51    07    E1     01D2   776          BBC     #7,R1,110$             ; IF BIT 7 CLEAR THEN DONE
                  51    E0 8F    8B     01D6   777          BICB3   #^XE0,R1,R0            ; REMOVE OTHER BITS
                           50            01DA
                     51    01    9A     01DB   778          MOVZBL  #1,R1                  ; SET ONE CHARACTER
                                 05     01DE   779 110$:     RSB                            ; RETURN
                                        01DF   780
                                        01DF   781          .END
```

```
ACB$B_RMOD                    0000000B     D
ACB$B_TYPE                    0000000A     D
ACB$C_LENGTH                  00000018     D
ACB$K_LENGTH                  00000018     D
ACB$L_AST                     00000010     D
ACB$L_ASTPRM                  00000014     D
ACB$L_ASTQBL                  00000004     D
ACB$L_ASTQFL                  00000000     D
ACB$L_KAST                    00000018     D
ACB$L_PID                     0000000C     D
ACB$V_QUOTA                 = 00000006     D
ACB$W_SIZE                    00000008     D
ACCESS                        0000013A R   D  01
CCB$B_AMOD                    00000009     D
CCB$B_STS                     00000008     D
CCB$C_LENGTH                  00000010     D
CCB$K_LENGTH                  00000010     D
CCB$L_DIRP                    0000000C     D
CCB$L_UCB                     00000000     D
CCB$L_WIND                    00000004     D
CCB$W_IOC                     0000000A     D
CCTABLE                       00000179 R   D  01
EXE$ABORTIO                   ********     X  01
EXE$CARRIAGE                  00000197 RG  D  01
EXE$C_CMSTKSZ               = 00000014     D
EXE$DEANONPAGED               ********     X  01
EXE$FINISHIO                  ********     X  01
EXE$MODIFY                    0000000C RG  D  01
EXE$MODIFYLOCK                00000044 RG  D  01
EXE$MODIFYLOCKR               00000047 RG  D  01
EXE$ONEPARM                   00000000 RG  D  01
EXE$QIODRVPKT                 ********     X  01
EXE$READ                      00000012 RG  D  01
EXE$READCHK                   000000CE RG  D  01
EXE$READCHKR                  000000E2 RG  D  01
EXE$READLOCK                  0000003E RG  D  01
EXE$READLOCKR                 00000051 RG  D  01
EXE$SENSEMODE                 0000016F RG  D  01
EXE$SETCHAR                   00000141 RG  D  01
EXE$SETMODE                   0000015F RG  D  01
EXE$WRITE                     0000001B RG  D  01
EXE$WRITECHK                  000000D4 RG  D  01
EXE$WRITECHKR                 0000010C RG  D  01
EXE$WRITELOCK                 00000041 RG  D  01
EXE$WRITELOCKR                00000058 RG  D  01
EXE$ZEROPARM                  00000006 RG  D  01
IO$_PHYSICAL                = 0000001F     D
IO$_READLBLK                = 00000021     D
IO$_READPBLK                = 0000000C     D
IO$_SETMODE                 = 00000023     D
IRP$B_CARCON                  00000038     D
IRP$B_EFN                     00000022     D
IRP$B_PRI                     00000023     D
IRP$B_RMOD                    0000000B     D
IRP$B_TYPE                    0000000A     D
IRP$C_LENGTH                  0000005C     D
IRP$K_LENGTH                  0000005C     D
IRP$L_ARB                     00000050     D
IRP$L_AST                     00000010     D
IRP$L_ASTPRM                  00000014     D
IRP$L_DIAGBUF                 00000044     D
IRP$L_EXTEND                  0000004C     D
IRP$L_IOQBL                   00000004     D
IRP$L_IOQFL                   00000000     D
IRP$L_IOSB                    00000024     D
IRP$L_IOST1                   00000034     D
IRP$L_IOST2                   00000038     D
IRP$L_MEDIA                   00000034     D
IRP$L_PID                     0000000C     D
IRP$L_SEGVBN                  00000040     D
IRP$L_SEQNUM                  00000048     D
IRP$L_SVAPTE                  0000002C     D
IRP$L_TT_TERM                 00000038     D
IRP$L_UCB                     0000001C     D
IRP$L_WIND                    00000018     D
IRP$M_FUNC                  = 00000002     D
IRP$Q_NT_PRVMSK               0000003C     D
IRP$S_FCODE                 = 00000006     D
IRP$V_DIAGBUF               = 00000007     D
IRP$V_FCODE                 = 00000000     D
IRP$V_FUNC                  = 00000001     D
IRP$W_ABCNT                   0000003C     D
IRP$W_BCNT                    00000032     D
IRP$W_BOFF                    00000030     D
IRP$W_CHAN                    00000028     D
IRP$W_FUNC                    00000020     D
IRP$W_OBCNT                   0000003E     D
IRP$W_SIZE                    00000008     D
IRP$W_STS                     0000002A     D
IRP$W_TT_PRMPT                0000003C     D
MMG$IOLOCK                    ********     X  01
P1                          = 00000000     D
P2                          = 00000004     D
P3                          = 00000008     D
P4                          = 0000000C     D
P5                          = 00000010     D
P6                          = 00000014     D
PCB$B_ASTACT                  0000000C     D
PCB$B_ASTEN                   0000000D     D
PCB$B_PRI                     0000000B     D
PCB$B_PRIB                    0000002F     D
PCB$B_TYPE                    0000000A     D
PCB$B_WEFC                    0000002E     D
PCB$C_LENGTH                  0000008C     D
PCB$K_LENGTH                  0000008C     D
PCB$L_ARB                     00000084     D
PCB$L_ASTQBL                  00000014     D
PCB$L_ASTQFL                  00000010     D
PCB$L_EFC2P                   00000058     D
PCB$L_EFC3P                   0000005C     D
PCB$L_EFCS                    00000050     D
PCB$L_EFCU                    00000054     D
PCB$L_EFWM                    0000004C     D
PCB$L_JIB                     00000078     D
```

PCB$L_OWNER		0000001C	D	UCB$B_TT_DETYPE		000000A4	D
PCB$L_PHD		00000064	D	UCB$B_TT_LFFILL		0000009E	D
PCB$L_PHYPCB		00000018	D	UCB$B_TT_SPEED		0000009C	D
PCB$L_PID		00000060	D	UCB$B_TYPE		0000000A	D
PCB$L_PQB		0000004C	D	UCB$B_VERTSZ		0000003F	D
PCB$L_SQBL		00000004	D	UCB$C_LENGTH		00000074	D
PCB$L_SQFL		00000000	D	UCB$C_MB_LENGTH		00000090	D
PCB$L_STS		00000024	D	UCB$C_TT_LENGTH		000000BC	D
PCB$L_UIC		00000088	D	UCB$K_LENGTH		00000074	D
PCB$L_WSSWP		00000020	D	UCB$K_MB_LENGTH		00000090	D
PCB$L_WTIME		00000028	D	UCB$K_TT_LENGTH		000000BC	D
PCB$Q_PRIV		0000007C	D	UCB$L_AMB		00000054	D
PCB$T_LNAME		00000068	D	UCB$L_ASTQBL		00000010	D
PCB$T_TERMINAL		00000044	D	UCB$L_ASTQFL		0000000C	D
PCB$W_APTCNT		00000030	D	UCB$L_CPID		0000005C	D
PCB$W_ASTCNT		00000038	D	UCB$L_CRB		00000020	D
PCB$W_BIOCNT		0000003A	D	UCB$L_DDB		00000024	D
PCB$W_BIOLM		0000003C	D	UCB$L_DEVCHAR		00000034	D
PCB$W_DIOCNT		0000003E	D	UCB$L_DEVDEPEND		0000003C	D
PCB$W_DIOLM		00000040	D	UCB$L_DPC		00000080	D
PCB$W_GPGCNT		00000034	D	UCB$L_DUETIM		0000005C	D
PCB$W_GRP		0000008A	D	UCB$L_DX_BFPNT		0000009C	D
PCB$W_MEM		00000088	D	UCB$L_DX_BUF		00000098	D
PCB$W_MTXCNT		0000000E	D	UCB$L_DX_RXDB		000000A0	D
PCB$W_PPGCNT		00000036	D	UCB$L_EMB		00000078	D
PCB$W_PRCCNT		00000042	D	UCB$L_FIRST		00000014	D
PCB$W_SIZE		00000008	D	UCB$L_FPC		0000000C	D
PCB$W_STATE		0000002C	D	UCB$L_FQBL		00000004	D
PCB$W_TMBU		00000032	D	UCB$L_FQFL		00000000	D
SIZ...	=	00000002	D	UCB$L_FR3		00000010	D
SS$_ACCVIO	=	0000000C	D	UCB$L_FR4		00000014	D
SS$_NORMAL	=	00000001	D	UCB$L_IOQBL		00000044	D
UCB$B_AMOD		00000053	D	UCB$L_IOQFL		00000040	D
UCB$B_CEX		00000077	D	UCB$L_IRP		0000004C	D
UCB$B_CM1		0000004A	D	UCB$L_LINK		0000002C	D
UCB$B_CM2		0000004B	D	UCB$L_LOGADR		00000064	D
UCB$B_DEVCLASS		00000038	D	UCB$L_MAXBLOCK		00000084	D
UCB$B_DEVTYPE		00000039	D	UCB$L_MB_MBX		0000007C	D
UCB$B_DIPL		00000052	D	UCB$L_MB_PORT		0000008C	D
UCB$B_DX_SCTCNT		000000A6	D	UCB$L_MB_RAST		00000078	D
UCB$B_ERTCNT		00000070	D	UCB$L_MB_SHB		00000080	D
UCB$B_ERTMAX		00000071	D	UCB$L_MB_WAST		00000074	D
UCB$B_FEX		00000076	D	UCB$L_MB_WIOQBL		00000088	D
UCB$B_FIPL		0000000B	D	UCB$L_MB_WIOQFL		00000084	D
UCB$B_LOCSRV		0000003C	D	UCB$L_MEDIA		0000008C	D
UCB$B_OFFNDX		00000094	D	UCB$L_NT_DATSSB		00000074	D
UCB$B_OFFRTC		00000095	D	UCB$L_NT_INTSSB		00000078	D
UCB$B_REMSRV		0000003D	D	UCB$L_OPCNT		00000060	D
UCB$B_SECTORS		0000003C	D	UCB$L_OWNUIC		0000001C	D
UCB$B_SLAVE		00000074	D	UCB$L_PID		00000028	D
UCB$B_SPR		00000075	D	UCB$L_RQBL		00000004	D
UCB$B_STATE		00000052	D	UCB$L_RQFL		00000000	D
UCB$B_TRACKS		0000003D	D	UCB$L_SVAPTE		00000068	D
UCB$B_TT_CRFILL		0000009D	D	UCB$L_SVPN		00000064	D
UCB$B_TT_DECRF		000000A1	D	UCB$L_TT_DECHAR		000000A8	D
UCB$B_TT_DELFF		000000A2	D	UCB$L_TT_RDUE		0000008C	D
UCB$B_TT_DESPEE		000000A0	D	UCB$L_TT_RTIMOU		000000B8	D

```
UCB$L_VCB                       00000030     D
UCB$T_PARTNER                   0000000C     D
UCB$W_BCNT                      0000006E     D
UCB$W_BCR                       00000096     D
UCB$W_BOFF                      0000006C     D
UCB$W_BUFQUO                    00000018     D
UCB$W_BYTESTOGO                 0000003E     D
UCB$W_CHARGE                    0000004A     D
UCB$W_CYLINDERS                 0000003E     D
UCB$W_DA                        0000008C     D
UCB$W_DC                        0000008E     D
UCB$W_DEVBUFSIZ                 0000003A     D
UCB$W_DEVSTS                    0000005A     D
UCB$W_DIRSEQ                    00000088     D
UCB$W_DSTADDR                   00000C018    D
UCB$W_DX_BCR                    000000A4     D
UCB$W_ECT                       00000090     D
UCB$W_EC2                       00000092     D
UCB$W_ERRCNT                    00000072     D
UCB$W_FUNC                      0000007E     D
UCB$W_MB_SEED                   00000000     D
UCB$W_MSGCNT                    00000016     D
UCB$W_MSGMAX                    00000014     D
UCB$W_NT_CHAN                   0000007C     D
UCB$W_OFFSET                    0000008A     D
UCB$W_REFC                      00000050     D
UCB$W_SIZE                      00000008     D
UCB$W_SRCADDR                   0000001A     D
UCB$W_STS                       00000058     D
UCB$W_TT_DESIZE                 000000A5     D
UCB$W_UNIT                      00000048     D
UCB$W_VPROT                     0000001A     D
VA$M_BYTE                     = 000001FF     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
$EP	000001DF (479.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	000000BC (188.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACB$V_QUOTA	=00000006		#-378 (1)
ACCESS	0000013A-R	563 (1)	#-508 (1) #-554 (1) #-601 (1) #-644 (1)
CCB$W_IOC	0000000A		#-371 (1)
CCTABLE	00000179-R	732 (1)	751 (1)
EXE$ABORTIO	00000000-XR		#-391 (1) #-464 (1)
EXE$CARRIAGE	00000197-R	748 (1)	
EXE$C_CMSTKSZ	=00000014	0.5 (1)	#-385 (1)
EXE$DEANONPAGED	00000000-XR		374.1 (1) 380.1 (1)
EXE$FINISHIO	00000000-XR		#-683 (1)
EXE$MODIFY	0000000C-R	191.2 (1)	
EXE$MODIFYLOCK	00000044-R	287 (1)	191.3 (1)
EXE$MODIFYLOCKR	00000047-R	342 (1)	#-288 (1)
EXE$ONEPARM	00000000-R	99.5 (1)	
EXE$QIODRVPKT	00000000-XR		#-158.3 (1) #-191.20 (1) #-645 (1)
EXE$READ	00000012-R	191.5 (1)	
EXE$READCHK	000000CE-R	425 (1)	
EXE$READCHKR	000000E2-R	502 (1)	#-344 (1) #-350 (1) #-427 (1)
EXE$READLOCK	0000003E-R	275 (1)	191.6 (1)
EXE$READLOCKR	00000051-R	348 (1)	#-276 (1)
EXE$SENSEMODE	0000016F-R	680 (1)	
EXE$SETCHAR	00000141-R	599 (1)	
EXE$SETMODE	0000015F-R	642 (1)	
EXE$WRITE	0000001B-R	191.8 (1)	
EXE$WRITECHK	000000D4-R	460 (1)	
EXE$WRITECHKR	0000010C-R	548 (1)	#-355 (1) #-462 (1)
EXE$WRITELOCK	00000041-R	281 (1)	191.9 (1)
EXE$WRITELOCKR	00000058-R	353 (1)	#-282 (1)
EXE$ZEROPARM	00000006-R	158.1 (1)	
IO$_PHYSICAL	=0000001F		#-191.12 (1)
IO$_READLBLK	=00000021		#-191.14 (1)
IO$_READPBLK	=0000000C		#-191.14 (1)
IO$_SETMODE	=00000023		#-602 (1)
IRP$B_CARCON	00000038		#-191.10 (1) #-749 (1) #-752 (1) #-761 (1) #-764 (1) #-765 (1) #-766 (1) #-769 (1) #-770 (1)
IRP$B_RMOD	0000000B		378 (1)
IRP$L_DIAGBUF	00000044		#-373 (1)
IRP$L_MEDIA	00000034		#-158.2 (1) #-645 (1) #-99.6 (1)
IRP$L_SVAPTE	0000002C		#-363 (1)
IRP$M_FUNC	=00000002		#-512 (1)
IRP$S_FCODE	=00000006		#-191.11 (1)
IRP$V_DIAGBUF	=00000007		#-372 (1)
IRP$V_FCODE	=00000000		#-191.11 (1)
IRP$V_FUNC	=00000001		#-191.7 (1)
IRP$W_BCNT	00000032		#-560 (1)
IRP$W_BOFF	00000030		#-358 (1)
IRP$W_FUNC	00000020		191.12 (1) #-191.15 (1)
IRP$W_STS	0000002A		191.7 (1) 372 (1) #-512 (1)
MMG$IOLOCK	00000000-XR		359.1 (1)
P1	=00000000	66 (1)	#-191.18 (1) #-600 (1) #-643 (1) #-99.6 (1)

```
P2                              =00000004      67      (1)     #-191.16 (1)
P3                              =00000008      68      (1)
P4                              =0000000C      69      (1)     #-191.10 (1)
P5                              =00000010      70      (1)
P6                              =00000014      71      (1)
PCB$W_ASTCNT                     00000038                      #-379    (1)
PCB$W_DIOCNT                     0000003E                      #-370    (1)
SS$_ACCVIO                      =0000000C                      #-565    (1)
SS$_NORMAL                      =00000001                      #-561    (1)     #-682    (1)
UCB$B_DEVCLASS                   00000038                      #-604    (1)
UCB$L_DEVDEPEND                  0000003C                      #-606    (1)     #-681    (1)
UCB$W_DEVBUFSIZ                  0000003A                      #-605    (1)
VA$M_BYTE                       =000001FF                      #-358    (1)     #-505    (1)     #-551    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	50 (1)	50 (1)
$CCBDEF	1	51 (1)	51 (1)
$DEFINI	1	50 (1)	50 (1) 51 (1) 52 (1) 53 (1) 54 (1) 55 (1) 56 (1) 57 (1) 58 (1)
$DEVDEF	1	52 (1)	52 (1)
$IODEF	17	53 (1)	53 (1)
$IRPDEF	4	54 (1)	54 (1)
$PCBDEF	4	55 (1)	55 (1)
$SSDEF	21	56 (1)	56 (1)
$UCBDEF	10	57 (1)	57 (1)
$VADEF	1	58 (1)	58 (1)
IFNORD	1	554 (1)	554 (1) 601 (1) 644 (1)
IFNOWRT	1	508 (1)	508 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.11	00:00:00.24
Command processing	142	00:00:00.87	00:00:02.02
Pass 1	691	00:00:16.56	00:00:23.84
Symbol table sort	0	00:00:01.91	00:00:04.28
Pass 2	184	00:00:03.73	00:00:14.79
Symbol table output	34	00:00:00.24	00:00:00.25
Psect synopsis output	7	00:00:00.02	00:00:00.02
Cross-reference output	29	00:00:00.33	00:00:00.75
Assembler run totals	1125	00:00:23.80	00:00:46.21

The working set limit was 1000 pages.
79127 bytes (155 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1212 non-local and 26 local symbols.
740 source lines were read in Pass 1, producing 0 object records in Pass 2.
57 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	6
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	3
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	16

1481 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) QIOFDT/UPDA=(QIOFDT.UPD,QIOFDT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 67 DECLARATIONS
(1) 121 QUEUE I/O REQUEST
(1) 548 COMPLETE I/O OPERATION
(1) 600 QUEUE I/O PACKET TO DRIVER
(1) 623 EXE$ALTQUEPKT - Call driver ALTSTART entry point
(1) 669 QUEUE I/O PACKET TO ACP
(1) 706 INSERT I/O PACKET IN UNIT QUEUE
(1) 729 INSERT I/O PACKET IN QUEUE BY PRIORITY

```
                        0000     .1          .TITLE  QIOREQ  *** QIOREQ handle QIO request
DS5.4                   0000     .2          .IDENT  /01-02/
02                      0000     .3          .NLIST  CND
02
-2                      0000      3
                        0000      4 ;
                        0000      5 ;*****************************************************************************
                        0000      6 ;*                                                                           *
DS5.4                   0000     .1 ; COPYRIGHT (C) 1977, 1980
-1                      0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000      9 ;*                                                                           *
                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                        0000     15 ;*  TRANSFERRED.                                                            *
                        0000     16 ;*                                                                           *
                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000     19 ;*  CORPORATION.                                                            *
                        0000     20 ;*                                                                           *
                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000     23 ;*                                                                           *
                        0000     24 ;*****************************************************************************
                        0000     25 ;
                        0000     26 ;++
                        0000     27 ;
                        0000     28 ; AUTHOR:
                        0000     29 ;
                        0000     30 ;       D. N. CUTLER,   14-JUN-76
                        0000     31 ;
                        0000     32 ; FACILITY:
                        0000     33 ;
                        0000     34 ;       SYSTEM SERVICE QUEUE I/O REQUEST
                        0000     35 ;
                        0000     36 ; MODIFIED BY:
DS5.4                   0000     .1 ;
DS5.4                   0000     .2 ;       DS5.4   John Ciukaj     5-May-1980      Version 5.4
DS5.4                   0000     .3 ;                 Used Vax/Vms version 2.0 SYSQIOREQ.MAR and created
DS5.4                   0000     .4 ;                 a UPD file with the edits necessary to reflect the
DS5.4                   0000     .5 ;                 changes exhibited in Supervisor version 5.3 module
DS5.4                   0000     .6 ;                 QIOREQ.
DS5.4                   0000     .7 ;                Dave Butenhof   30-jun-1980,    version 5.5
DS5.4                   0000     .8 ;       02        Alter some word displacement BSB's to longword displacement
DS5.4                   0000     .9 ;                 JSB's, since supervisor has expanded.
DS5.4                   0000    .10 ;
                        0000     37 ;
                        0000     38 ;       V0208   LMK0001         LEN KAWELL              03-OCT-1979
                        0000     39 ;               DISALLOW LOGICAL/PHYSICAL I/O TO A SPOOLED DEVICE UNLESS
                        0000     40 ;               THE PROCESS HAS LOG_IO PRIVILEGE.
                        0000     41 ;
                        0000     42 ;       V0207   KDM0053         Kathleen D. Morse       31-Aug-1979
                        0000     43 ;               ADD TWO NEW ENTRY POINTS FOR SHARED MEMORY GLOBAL SECTION
                        0000     44 ;               I/O, EXE$BLDPKTGSR AND EXE$BLDPKTGSW.
                        0000     45 ;
                        0000     46 ;       V0006   CHP0002         CAROL PETERS            14-AUG-1979
```

```
                                ;               IN EXE$ALTQUEPKT, IF NO ALTERNATE START I/O ADDRESS
                                ;               IS SPECIFIED IN THE DDR, JUST RETURN TO CALLER.
                                ;
                                ;       V0005   ACG0047         ANDREW C. GOLDSTEIN     09-AUG-1979
                                ;               ADD ACCESS RIGHTS BLOCK, PROTECTION INTERFACE CHANGES.
                                ;
                                ;       V04     CHP0001         CAROL PETERS            03-AUG-1979
                                ;               ADDED NEW ENTRY POINT FOR QUEUING A PACKET TO A DRIVER.
                                ;               ENTRY POINT IS STORED IN DDT AND CALLED ALTSTART.  ADDED
                                ;               SYSQIOREQ ENTRY POINT, EXE$ALTQUEPKT, WHICH JSBS TO
                                ;               THE ALTSTART ENTRY POIN, AND THEN RSBS TO ITS CALLER.
                                ;
                                ;       V03     RIH23909        R. I. HUSTVEDT          14-MAY-1979
                                ;               REMOVE SUPERFLUOUS BRANCH FOLLOWING CALL TO SCH$CLREF.
                                ;
                                ;       V02     PHL0002         P. H. LIPMAN            01-MAY-1978
                                ;               REWROTE BUILDPKT FOR SEGMENTED VIRTUAL I/O.
                                ;
                                ;--
```

```
                        0000     67          .SBTTL  DECLARATIONS
                        0000     68
                        0000     69 ;
                        0000     70 ; MACRO LIBRARY CALLS
                        0000     71 ;
                        0000     72
                        0000     73          $ACBDEF                         ;DEFINE ACB OFFSETS
                        0000     74          $AQBDEF                         ;DEFINE AQB OFFSETS
                        0000     75          $CADEF                          ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS
                        0000     76          $CCBDEF                         ;DEFINE CCB OFFSETS
                        0000     77          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000     78          $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000     79          $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                        0000     80          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000     81          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000     82          $PCBDEF                         ;DEFINE PCB OFFSETS
                        0000     83          $PHDDEF                         ;DEFINE PHD OFFSETS
                        0000     84          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000     85          $PRIDEF                         ;DEFINE PRIORITY CLASS INCREMENTS
                        0000     86          $PRVDEF                         ;DEFINE PRIVILEGE BITS
                        0000     87          $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                        0000     88          $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000     89          $SECDEF                         ;DEFINE SEC OFFSETS
                        0000     90          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000     91          $VCBDEF                         ;DEFINE VCB OFFSETS
                        0000     92          $WCBDEF                         ;DEFINE WINDOW CONTROL BLOCK OFFSETS
                        0000     93
                        0000     94 ;
                        0000     95 ; LOCAL SYMBOLS
                        0000     96 ;
                        0000     97 ; ARGUMENT LIST OFFSET DEFINITIONS
                        0000     98 ;
                        0000     99
                  00000004  0000    100 EFN=4                                   ;EVENT FLAG NUMBER
                  00000008  0000    101 CHAN=8                                  ;I/O CHANNEL NUMBER
                  0000000C  0000    102 FUNC=12                                 ;I/O FUNCTION CODE
                  00000010  0000    103 IOSB=16                                 ;ADDRESS OF I/O STATUS BLOCK
                  00000014  0000    104 ASTADR=20                               ;ADDRESS OF AST SERVICE ROUTINE
                  00000018  0000    105 ASTPRM=24                               ;AST SERVICE ROUTINE PARAMETER
                  0000001C  0000    106 P1=28                                   ;FIRST FUNCTION DEPENDENT PARAMETER
                  00000020  0000    107 P2=32                                   ;SECOND FUNCTION DEPENDENT PARAMETER
                  00000024  0000    108 P3=36                                   ;THIRD FUNCTION DEPENDENT PARAMETER
                  00000028  0000    109 P4=40                                   ;FOURTH FUNCTION DEPENDENT PARAMETER
                  0000002C  0000    110 P5=44                                   ;FIFTH FUNCTION DEPENDENT PARAMETER
                  00000030  0000    111 P6=48                                   ;SIXTH FUNCTION DEPENDENT PARAMETER
                        0000    112
                        0000    113 ;
                        0000    114 ; FUNCTION DECISION TABLE OFFSET DEFINITIONS
                        0000    115 ;
                        0000    116
                  00000000  0000    117 LEGAL=0                                 ;LEGAL FUNCTION MASK
                  00000008  0000    118 IOTYPE=8                                ;I/O FUNCTION TYPE MASK
                  00000010  0000    119 FDTACT=16                               ;ACTION ROUTINE MASKS
DS5.4                   00000000     .1          .PSECT  SEP, SHR, EXE, WRT, LONG
DS5.4                         0000   .2          MODNAM  QIOREQ
```

```
0007   121          .SBTTL  QUEUE I/O REQUEST
0007   122 ;+
0007   123 ; EXE$QIOREQ - QUEUE I/O REQUEST
0007   124 ;
0007   125 ; THIS SERVICE PROVIDES THE CAPABILITY TO INITIATE AN I/O OPERATION
0007   126 ; BY QUEUEING A REQUEST TO A DEVICE'S ASSOCIATED DRIVER.  ONCE THE
0007   127 ; OPERATION HAS BEEN INITIATED, CONTROL WILL RETURN TO THE CALLER
0007   128 ; WHO CAN SYNCHRONIZE I/O COMPLETION IN ONE OF THREE WAYS:
0007   129 ;
0007   130 ;       1) SPECIFY THE ADDRESS OF AN AST ROUTINE THAT WILL BE
0007   131 ;          EXECUTED WHEN THE I/O COMPLETES.
0007   132 ;
0007   133 ;       2) WAIT FOR THE SPECIFIED EVENT FLAG TO BE SET.
0007   134 ;
C007   135 ;       3) POLL THE SPECIFIED I/O STATUS BLOCK FOR A COMPLETION
0007   136 ;          STATUS.
0007   137 ;
0007   138 ; THIS ROUTINE VERIFYS THE FUNCTION INDEPENDENT PARAMETERS, ALLOCATES
0007   139 ; AN I/O REQUEST PACKET, COPIES THE FUNCTION INDEPENDENT PARAMETERS AND
0007   140 ; PROCESS INFORMATION TO THE I/O PACKET, CHECKS ACCESS TO THE DEVICE,
0007   141 ; AND CALLS THE DRIVER'S FUNCTION DECISION TABLE ROUTINE(S) THAT CORRESPOND
0007   142 ; TO THE SPECIFIED FUNCTION.  IT IS THEN UP TO THE FDT ROUTINE TO EITHER
0007   143 ; COMPLETE THE REQUEST IMMEDIATELY (EXE$ABORTIO OR EXE$FINISHIO) OR TO
0007   144 ; QUEUE THE I/O REQUEST FOR FURTHER PROCESSING BY THE DRIVER'S STARTIO
0007   145 ; ROUTINE (EXE$QIODRVPKT).
0007   146 ;
0007   147 ; INPUTS:
0007   148 ;
0007   149 ;       EFN(AP) = EVENT FLAG NUMBER.
0007   150 ;       CHAN(AP) = I/O CHANNEL NUMBER.
0007   151 ;       FUNC(AP) = I/O FUNCTION CODE.
C007   152 ;       IOSB(AP) = ADDRESS OF I/O STATUS BLOCK.
0007   153 ;       ASTADR(AP) = ADDRESS OF AST SERVICE ROUTINE.
0007   154 ;       ASTPRM(AP) = AST SERVICE ROUTINE PARAMETER.
0007   155 ;       P1(AP) TO P6(AP) = FUNCTION DEPENDENT PARAMETERS.
0007   156 ;
C007   157 ;       R4 = CURRENT PROCESS PCB ADDRESS.
0007   158 ;
0007   159 ; OUTPUTS:
0007   160 ;
0007   161 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO INITIATE THE I/O REQUEST.
0007   162 ;
0007   163 ;               R0 = SS$_ABORT - A NETWORK LOGICAL LINK WAS BROKEN.
0007   164 ;
0007   165 ;               R0 = SS$_ACCVIO - THE I/O STATUS BLOCK CANNOT BE WRITTEN BY
C007   166 ;                    THE CALLER.
0007   167 ;
0007   168 ;               R0 = SS$_DEVOFFLINE - THE SPECIFIED DEVICE IS OFFLINE.
0007   169 ;
0007   170 ;               R0 = SS$_EXQUOTA - THE PROCESS HAS EXCEEDED ITS BUFFERED I/O
0007   171 ;                    QUOTA, DIRECT I/O QUOTA, OR BUFFERED I/O BYTE COUNT
0007   172 ;                    QUOTA AND HAS DISABLED RESOURCE WAIT MODE.  OR, THE
0007   173 ;                    PROCESS HAS EXCEEDED ITS AST LIMIT QUOTA.
0007   174 ;
0007   175 ;               R0 = SS$_ILLEFC - AN ILLEGAL EVENT FLAG NUMBER WAS SPECIFIED.
0007   176 ;
0007   177 ;               R0 = SS$_INSFMEM - INSUFFICIENT DYNAMIC MEMORY IS AVAILABLE
```

```
                                0007   178 ;                   TO ALLOCATE AN I/O REQUEST PACKET AND THE PROCESS HAS
                                0007   179 ;                   DISABLED RESOURCE WAIT MODE.
                                0007   180 ;
                                0007   181 ;           R0 = SS$_IVCHAN - AN INVALID CHANNEL NUMBER WAS SPECIFIED.
                                0007   182 ;
                                0007   183 ;           R0 = SS$_NOPRIV - THE SPECIFIED CHANNEL DOES NOT EXIST OR WAS
                                0007   184 ;                   ASSIGNED FROM A MORE PRIVILEGED ACCESS MODE.  OR, THE
                                0007   185 ;                   PROCESS DOES NOT HAVE THE PRIVILEGE TO PERFORM THE
                                0007   186 ;                   SPECIFIED TYPE OF I/O FUNCTION ON THE DEVICE.
                                0007   187 ;
                                0007   188 ;           R0 = SS$_UNASEFC - UNASSOCIATED EVENT FLAG CLUSTER SPECIFIED.
                                0007   189 ;
                                0007   190 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0007   191 ;
                                0007   192 ;           R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0007   193 ;-
                                0007   194
                                0007   195          .ENABL  LSB
DS5.4                     0FFC  0007    .1          .ENTRY  VMS$QIO,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
DS5.4          00000000'EF  DE  0009    .2          MOVAL   DS$AX_SOFTPCB, R4       ; Get PCB address into R4
                        54      000F
DS5.4           7E  04 AC  9A  0010    .3 10$:     MOVZBL  EFN(AP),-(SP)           ;GET EVENT FLAG #
DS5.4                 01   FB  0014    .4          CALLS   #1,@#SYS$CLREF          ;CLEAR SPECIFIED EVENT FLAG
                00000000'9F       0016
DS5.4              03 50   E8  001B    .5          BLBS    R0, 15$                 ; Event flag cleared OK
DS5.4                0080  31  001E    .6          BRW     ERROR                   ; Didn't work
DS5.4                          0021    .7 15$:
-3                 50  08 AC  3C  0021  199         MOVZWL  CHAN(AP),R0             ;GET CHANNEL NUMBER
                       FFD8'  30  0025  200         BSBW    IOC$VERIFYCHAN          ;VERIFY CHANNEL NUMBER
                   56  51     D0  0028  201         MOVL    R1,R6                   ;SAVE ADDRESS OF CCB
                     73 50     E9  002B  202         BLBC    R0,ERROR                ;IF LBC INVALID CHANNEL
                   55  66     D0  002E  203         MOVL    CCB$L_UCB(R6),R5        ;GET ASSIGNED DEVICE UCB ADDRESS
                   59  52     D0  0031  204         MOVL    R2,R9                   ;SAVE CHANNEL INDEX
                5A  0C AC     3C  0034  205         MOVZWL  FUNC(AP),R10            ;GET I/O FUNCTION CODE AND MODIFIERS
              FFFFFFFF'8F     CB  0038  206         BICL3   #^C<IO$M_FCODE>,R10,R7  ;CLEAR ALL BUT I/O FUNCTION CODE
                   57  5A           003E
           09 34 A5  00'      E1  0040  207         BBC     S^#DEV$V_SPL,UCB$L_DEVCHAR(R5),17$ ;IF CLR, DEVICE NOT SPOOLED
                   57  00'     D1  0045  208         CMPL    S^#IO$_LOGICAL,R7       ;VIRTUAL I/O FUNCTION?
                       04     18  0048  209         BGEQ    17$                     ;IF GEQ NO
                55  54 A5     D0  004A  210         MOVL    UCB$L_AMB(R5),R5        ;GET INTERMEDIATE DEVICE UCB ADDRESS
                50  24 A5     D0  004E  211 17$:    MOVL    UCB$L_DDB(R5),R0        ;GET ADDRESS OF DDB
                50  0C A0     D0  0052  212         MOVL    DDB$L_DDT(R0),R0        ;GET ADDRESS OF DDT
DS5.4           58  08 A0     D0  0056    .1        MOVL    DDT$L_FDT(R0),R8        ;GET ADDRESS OF FDT
DS5.4           03 58   10    E0  005A    .2        BBS     #16,R8,20$              ;BRANCH IF SUPERVISOR ADDRESS
DS5.4              58   50    C0  005E    .3        ADDL    R0,R8                   ;CHANGE OFFSET TO ABSOLUTE
-3              2B 68   57    E1  0061  216 20$:    BBC     R7,LEGAL(R8),50$        ;IF CLR, ILLEGAL I/O FUNCTION
             2D 58 A5   04    E1  0065  217         BBC     #UCB$V_ONLINE,UCB$W_STS(R5),60$ ;IF CLR, DEVICE OFFLINE
                51  10 AC     D0  006A  218         MOVL    IOSB(AP),R1             ;GET ADDRESS OF I/O STATUS BLOCK
                       08     13  006E  219         BEQL    30$                     ;IF EQL NONE SPECIFIED
                               0070  220         IFNOWRT #8,(R1),70$             ;CAN I/O STATUS BLOCK BE WRITTEN?
                       61     7C  0076  221         CLRQ    (R1)                    ;CLEAR I/O STATUS BLOCK
                       7E     DC  0078  222 30$:    MOVPSL  -(SP)                   ;READ CURRENT PSL
                5B  3A A4     DE  007A  223         MOVAL   PCB$W_BIOCNT(R4),R11    ;SET FOR BUFFERED I/O FUNCTION
             04 08 A8  57     E0  007E  224         BBS     R7,IOTYPE(R8),40$       ;IF SET, BUFFERED I/O FUNCTION
                5B  3E A4     DE  0083  225         MOVAL   PCB$W_DIOCNT(R4),R11    ;SET FOR DIRECT I/O FUNCTION
                   52  5B     D0  0087  226 40$:    MOVL    R11,R2                  ;SET ADDRESS OF USAGE COUNT WORD
-2                    31 04 A6  E9  008A  229         BLBC    CCB$L_WIND(R6),GTPKT    ;IF LBC ACCESS/DEACCESS NOT PENDING
```

```
DS5.4             80   11  008E    .1          BRB     10$                         ;
-1       50  0000'8F   3C  0090   231 50$:     MOVZWL  #SS$_ILLIOFUNC,R0           ;SET ILLEGAL I/O FUNCTION STATUS
                  0A   11  0095   232          BRB     ERROR                       ;
         50  0000'8F   3C  0097   233 60$:     MOVZWL  #SS$_DEVOFFLINE,R0          ;SET DEVICE OFFLINE STATUS
                  03   11  009C   234          BRB     ERROR                       ;
             50   00'  3C  009E   235 70$:     MOVZWL  S^#SS$_ACCVIO,R0            ;SET ACCESS VIOLATION STATUS
                           00A1   236 ERROR:   SETIPL  #0                          ;ALLOW INTERRUPTS
                  50   DD  00A4   237          PUSHL   R0                          ;SAVE FINAL STATUS
DS5.4      7E  04 AC   9A  00A6    .1          MOVZBL  EFN(AP),-(SP)               ;GET SPECIFIED EVENT FLAG #
DS5.4             01   FB  00AA    .2          CALLS   #1,@#SYS$SETEF              ;SET SPECIFIED EVENT FLAG
          00000000'9F      00AC
-4                01   BA  00B1   242          POPR    #^M<R0>                     ;RESTORE FINAL STATUS
                       04  00B3   243          RET                                 ;
-7                         00B4   251
                           00B4   252 ;
                           00B4   253 ; ALLOCATE REQUEST I/O PACKET
                           00B4   254 ;
                           00B4   255
02        00000000'EF  16  00B4    .1 ALLOC:   JSB     L^EXE$ALLOCIRP              ; Allocate I/O request packet
-1              0B 50  E8  00BA   257          BLBS    R0,SUCCES                   ;IF LBS SUCCESSFUL ALLOCATION
                  E2   11  00BD   258          BRB     ERROR                       ;
02        00000000'FF  0F  00BF    .1 GTPKT:   REMQUE  @L^IOC$GL_IRPFL, R2         ; Get I/O packet from lookaside list
                  52       00C5
-1                EC   1D  00C6   260          BVS     ALLOC                       ;IF VS EMPTY LIST
                           00C8   261          .DSABL  LSB
                           00C8   262
                           00C8   263 ;
                           00C8   264 ; BUILD DEVICE INDEPENDENT PART OF I/O PACKET
                           00C8   265 ;
                           00C8   266
                  6B   B7  00C8   267 SUCCES:  DECW    (R11)                       ;UPDATE I/O COUNT FOR FUNCTION TYPE
               0A A6   B6  00CA   268          INCW    CCB$W_IOC(R6)               ;INCREMENT OUTSTANDING I/O ON CHANNEL
              53  52   D0  00CD   269          MOVL    R2,R3                       ;COPY ADDRESS OF ALLOCATED I/O PACKET
              52  08   C0  00D0   270          ADDL    #IRP$W_SIZE,R2              ;POINT TO SIZE FIELD
         82 005C  8F   B0  00D3   271          MOVW    #IRP$C_LENGTH,(R2)+         ;INSERT LENGTH OF I/O PACKET
              82  0A   90  00D8   272          MOVB    #DYN$C_IRP,(R2)+            ;INSERT DATA STRUCTURE TYPE
         6E   02  16   EF  00DB   273          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,(SP),(R2) ;INSERT ACCESS MODE
                  62       00DF
                  52   D6  00E0   274          INCL    R2                          ;ADJUST PAST ACCESS MODE FIELD
            82 60 A4   D0  00E2   275          MOVL    PCB$L_PID(R4),(R2)+         ;INSERT PROCESS ID OF CURRENT PROCESS
            82 14 AC   7D  00E6   276          MOVQ    ASTADR(AP),(R2)+            ;INSERT AST ADDRESS AND PARAMETER
DS5.4       82 04 A6   D0  00EA    .1          MOVL    CCB$L_WIND(R6),(R2)+        ;Insert Window address
-7            82  55   D0  00EE   284          MOVL    R5,(R2)+                    ;INSERT DEVICE UCB ADDRESS
              82  5A   B0  00F1   285          MOVW    R10,(R2)+                   ;INSERT I/O FUNCTION CODE
            82 04 AC   90  00F4   286          MOVB    EFN(AP),(R2)+               ;INSERT EVENT FLAG NUMBER
DS5.4             82   94  00F8    .1          CLRB    (R2)+                       ;INSERT PROCESS BASE PRIORITY
-1          82 10 AC   D0  00FA   288          MOVL    IOSB(AP),(R2)+              ;INSERT I/O STATUS BLOCK ADDRESS
              82  59   3C  00FE   289          MOVZWL  R9,(R2)+                    ;INSERT CHANNEL INDEX AND ZERO STATUS
         04 08 A8 57   E1  0101   290          BBC     R7,IOTYPE(R8),20$           ;IF CLR, DIRECT I/O FUNCTION
            FE A2 01   A8  0106   291          BISW    #IRP$M_BUFIO,-2(R2)         ;SET TO BUFFERED I/O FUNCTION
                  62   7C  010A   292 20$:     CLRQ    (R2)                        ;CLEAR PTE ADDRESS, BYTE OFFSET, AND BYTE CO
-7                         010C   300
                           010C   301 ;
                           010C   302 ; CHECK IF REQUESTING PROCESS HAS PRIVILEGE TO ACCESS DEVICE
                           010C   303 ;
                           010C   304
              57  00'  D1  010C   305 ACCESS:  CMPL    S^#IO$_WRITEVBLK,R7         ;POSSIBLY VIRTUAL READ OR WRITE?
```

```
                     20   1A  010F   306          BGTRU   15$                             ;IF GTRU NO
                 57  00'  D1  0111   307          CMPL    S^#IO$_READVBLK,R7              ;VIRTUAL READ OR WRITE?
                     1B   1F  0114   308          BLSSU   15$                             ;IF LSSU NO
              0F 34 A5  00'  E1  0116   309          BBC     S^#DEV$V_FOD,UCB$L_DEVCHAR(R5),5$ ;IF CLR, NOT FILE DEVICE
                             011B   310
                             011B   311 ;
                             011B   312 ; THE FOLLOWING TEST IS NECESSITATED BY THE SYSTEM INITIALIZATION SEQUENCE
                             011B   313 ;
                             011B   314
                    18 A3    D5  011B   315          TSTL    IRP$L_WIND(R3)          ;WINDOW ADDRESS SPECIFIED?
                     11      12  011E   316          BNEQ    15$                     ;IF NEQ YES
              0E 34 A5  00'  E1  0120   317          BBC     S^#DEV$V_MNT,UCB$L_DEVCHAR(R5),60$ ;IF CLR, DEVICE NOT MOUNTED
              0C 34 A5  00'  E1  0125   318          BBC     S^#DEV$V_FOR,UCB$L_DEVCHAR(R5),80$ ;IF CLR, MOUNTED STRUCTURED
                             012A   319
                             012A   320 ;
                             012A   321 ; CONVERT VIRTUAL READ/WRITE FUNCTION TO ITS LOGICAL COUNTERPART
                             012A   322 ;
                             012A   323
                 57  00'  C2  012A   324 5$:     SUBL    S^#IO$_READVBLK-IO$_READLBLK,R7 ;CONVERT TO LOGICAL FUNCTION
                20 A3  00'  A2  012D   325          SUBW    S^#IO$_READVBLK-IO$_READLBLK,IRP$W_FUNC(R3) ;
DS5.4                03   11  0131    .1 15$:     BRB     80$                     ; Don't care about VIRTUAL/LOG/PHYS rights
DS5.4                            0133    .2 60$:
-37                0064   31  0133   363 70$:     BRW     EXE$ABORTIO             ;
                             0136   364
                             0136   365 ;
                             0136   366 ; PROCESS HAS ACCESS TO DEVICE
                             0136   367 ;
                             0136   368
                 57  00'  D1  0136   369 80$:     CMPL    S^#IO$_PHYSICAL,R7      ;LOGICAL OR VIRTUAL I/O FUNCTION?
                     39   19  0139   370          BLSS    90$                     ;IF LSS YES
                0100 8F   A8  013B   371          BISW    #IRP$M_PHYSIO,IRP$W_STS(R3) ;SET PHYSICAL I/O FLAG
                   2A A3        013F
              59    30 AC   D0  0141   372          MOVL    P6(AP),R9               ;GET ADDRESS OF DIAGNOSTIC BUFFER
                     2D   13  0145   373          BEQL    90$                     ;IF EQL NONE SPECIFIED
-1            51    24 A5   D0  0147   375          MOVL    UCB$L_DDB(R5),R1        ;GET ADDRESS OF DDB
              51    0C A1   D0  014B   376          MOVL    DDB$L_DDT(R1),R1        ;GET ADDRESS OF DDT
              51    14 A1   3C  014F   377          MOVZWL  DDT$W_DIAGBUF(R1),R1    ;GET SIZE OF DIAGNOSTIC BUFFER
                     1F   13  0153   378          BEQL    90$                     ;IF EQL NO DIAGNOSTIC FUNCTIONS
                     53   DD  0155   379          PUSHL   R3                      ;SAVE I/O PACKET ADDRESS
02                00000000'EF   16  0157    .1     JSB     L^EXE$ALLOCBUF          ; Allocate diagnostic buffer
-1                        53 8ED0  015D   381          POPL    R3                      ;RETRIEVE I/O PALKET ADDRESS
                  D0  50   E9  0160   382          BLBC    R0,70$                  ;IF LBC ALLOCATION FAILURE
              44 A3   52   D0  0163   383          MOVL    R2,IRP$L_DIAGBUF(R3)    ;SAVE ADDRESS OF DIAGNOSTIC BUFFER
              82   0C A2   9E  0167   384          MOVAB   12(R2),(R2)+            ;SET POINTER TO DATA AREA
                 62   59   D0  016B   385          MOVL    R9,(R2)                 ;SAVE USER ADDRESS OF DIAGNOSTIC BUFFER
                0080 8F   A8  016E   386          BISW    #IRP$M_DIAGBUF,IRP$W_STS(R3) ;SET DIAGNOSTIC BUFFER PRESENT
                   2A A3        0172
                             0174   387
                             0174   388 ;
                             0174   389 ; CHECK IF AST IS SPECIFIED
                             0174   390 ;
                             0174   391
                    10 A3    D5  0174   392 90$:     TSTL    IRP$L_AST(R3)           ;AST ADDRESS SPECIFIED?
                     05      13  0177   393          BEQL    100$                    ;IF EQL NO
-4                 0B A3   40 8F   88  0179   398          BISB    #ACB$M_QUOTA,IRP$B_RMOD(R3) ;SET AST QUOTA UPDATE FLAG
                             017E   399
                             017E   400 ;
```

```
                                017E   401 ; SCAN FUNCTION DECISION TABLE CALLING EACH SELECTED ACTION ROUTINE WITH:
                                017E   402 ;
                                017E   403 ;       R0 = SCRATCH.
                                017E   404 ;       R1 = SCRATCH.
                                017E   405 ;       R2 = SCRATCH.
                                017E   406 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                017E   407 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                017E   408 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                017E   409 ;       R6 = ADDRESS OF CCB.
                                017E   410 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                017E   411 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                017E   412 ;       R9 = SCRATCH.
                                017E   413 ;       R10 = SCRATCH.
                                017E   414 ;       R11 = SCRATCH.
                                017E   415 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                017E   416 ;
                                017E   417
                    58    04    C0  017E   418 100$:   ADDL    #FDTACT-12,R8          ;POINT TO ACTION ROUTINE MASKS
                    5C    1C    C0  0181   419         ADDL    #P1,AP                 ;POINT TO FIRST FUNCTION DEPENDENT PARAMETER
                    58    0C    C0  0184   420 110$:   ADDL    #12,R8                 ;POINT TO NEXT FUNCTION MASK
                 F9 68    57    E1  0187   421         BBC     R7,(R8),110$           ;IF CLR, THEN ACTION NOT SELECTED
DS5.4               50      08 A8  D0  018B    .1         MOVL    8(R8),R0               ;GET ADDRESS OF ACTION ROUTINE
DS5.4               03 50    10    E0  018F    .2         BBS     #16,R0,120$            ;BRANCH IF ABSOLUTE SUPERVISOR ADDR
DS5.4                  50    58    C0  0193    .3         ADDL    R8,R0                  ;CHANGE OFFSET TO ABSOLUTE
-3                           60    16  0196   425 120$:   JSB     (R0)                   ;CALL ACTION ROUTINE
                             EA    11  0198   426         BRB     110$                   ;
```

```
                        019A   548          .SBTTL  COMPLETE I/O OPERATION
                        019A   549 ;+
                        019A   550 ; EXE$ABORTIO - ABORT I/O OPERATION
                        019A   551 ;
                        019A   552 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                        019A   553 ; TO FINISH AN I/O OPERATION WITHOUT RETURNING THE FINAL I/O STATUS.
                        019A   554 ;
                        019A   555 ; EXE$FINISHIO - FINISH I/O OPERATION
                        019A   556 ;
                        019A   557 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                        019A   558 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS.
                        019A   559 ;
                        019A   560 ; EXE$FINISHIOC - FINISH I/O OPERATION WITH SECOND I/O STATUS LONGWORD CLEARED
                        019A   561 ;
                        019A   562 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DESCISION TABLE ACTION ROUTINE
                        019A   563 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS WITH THE
                        019A   564 ; SECOND I/O STATUS LONGWORD CLEARED.
                        019A   565 ;
                        019A   566 ; INPUTS:
                        019A   567 ;
                        019A   568 ;       R0 = FIRST LONGWORD OF FINAL I/O STATUS.
                        019A   569 ;       R1 = SECOND LONGWORD OF FINAL I/O STATUS.
                        019A   570 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        019A   571 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        019A   572 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                        019A   573 ;
                        019A   574 ; OUTPUTS:
                        019A   575 ;
                        019A   576 ;       THE FINAL I/O STATUS IS STORED IN THE I/O PACKET AND THE PACKET IS
                        019A   577 ;       INSERTED IN THE I/O POST PROCESSING QUEUE. A SOFTWARE INTERRUPT
                        019A   578 ;       IS GENERATED TO INITIATE I/O POST PROCESSING AND THE FIRST WORD
                        019A   579 ;       OF THE FINAL I/O STATUS IS RETURNED AS THE SERVICE STATUS.
                        019A   580 ;-
                        019A   581
                        019A   582          .ENABL  LSB
                        019A   583 EXE$ABORTIO::                                ;ABORT I/O OPERATION
          24 A3   D4    019A   584          CLRL    IRP$L_IOSB(R3)              ;CLEAR ADDRESS OF I/O STATUS BLOCK
    11 0B A3   06   E5  019D   585          BBCC    #ACB$V_QUOTA,IRP$B_RMOD(R3),10$ ;IF CLR, NO AST SPECIFIED
          38 A4   B6    01A2   586          INCW    PCB$W_ASTCNT(R4)            ;UPDATE AVAILABLE AST QUEUE ENTRIES
             0C   11    01A5   587          BRB     10$                         ;
                        01A7   588 EXE$FINISHIOC::                              ;FINISH I/O OPERATION CLEAR SECOND LONGWORD
             51   D4    01A7   589          CLRL    R1                          ;CLEAR SECOND I/O STATUS LONGWORD
                        01A9   590 EXE$FINISHIO::                               ;FINISH I/O OPERATION
          60 A5   D6    01A9   591          INCL    UCB$L_OPCNT(R5)             ;INCREMENT OPERATIONS COMPLETED
       34 A3   50   7D  01AC   592          MOVQ    R0,IRP$L_MEDIA(R3)          ;STORE FINAL I/O STATUS
          50   00'  3C  01B0   593          MOVZWL  S^#SS$_NORMAL,R0            ;SET NORMAL COMPLETION STATUS
DS5.4        63   0E    01B3     .1 10$:    INSQUE  (R3),@IOC$GL_PSBL           ;INSERT I/O PACKET IN POST PROCESS QUEUE
     00000000'FF        01B5
-1           42   12    01BA   595          BNEQ    QIORETURN                   ;IF NEQ NOT FIRST ENTRY IN QUEUE
                        01BC   596          SOFTINT #IPL$_IOPOST                ;INITIATE SOFTWARE INTERRUPT
             3D   11    01BF   597          BRB     QIORETURN                   ;
                        01C1   598          .DSABL  LSB
```

```
                     01C1   600          .SBTTL  QUEUE I/O PACKET TO DRIVER
                     01C1   601 ;+
                     01C1   602 ; EXE$QIODRVPKT - QUEUE I/O PACKET TO DRIVER
                     01C1   603 ;
                     01C1   604 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                     01C1   605 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE DRIVER.
                     01C1   606 ;
                     01C1   607 ; INPUTS:
                     01C1   608 ;
                     01C1   609 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     01C1   610 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     01C1   611 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     01C1   612 ;
                     01C1   613 ; OUTPUTS:
                     C1C1   614 ;
                     01C1   615 ;       THE I/O PACKET IS QUEUED BY PRIORITY IN THE APPROPRIATE DEVICE
                     01C1   616 ;       QUEUE AND A NORMAL COMPLETION STATUS IS RETURNED.
                     01C1   617 ;-
                     01C1   618
                     01C1   619 EXE$QIODRVPKT::                          ;QUEUE I/O PACKET
3F   10              01C1   620          BSBB    EXE$INSIOQ             ;INSERT I/O PACKET IN DEVICE QUEUE
34   11              01C3   621          BRB     EXE$QIORETURN          ;
```

```
                                                01C5   623          .SBTTL  EXE$ALTQUEPKT - Call driver ALTSTART entry point
                                                01C5   624
                                                01C5   625 ;
                                                01C5   626 ; EXE$ALTQUEPKT - activates a driver at its ALTSTART entry point
                                                01C5   627 ;
                                                01C5   628 ; Routine description:
                                                01C5   629 ;
                                                01C5   630 ;       Locates and calls a driver entry point supplied as an alternate
                                                01C5   631 ;       START I/O entry point. Does not test for unit busy before the
                                                01C5   632 ;       call. Exits by returning to caller.
                                                01C5   633 ;
                                                01C5   634 ;       The routine expects to gain control at or below driver fork
                                                01C5   635 ;       level. The routine raises to driver fork IPL before the call,
                                                01C5   636 ;       and restores the previous IPL before returning to its caller.
                                                01C5   637 ;
                                                01C5   638 ; Inputs:
                                                01C5   639 ;
                                                01C5   640 ;       R3      - address of packet or buffer
                                                01C5   641 ;       R5      - address of UCB
                                                01C5   642 ;
                                                01C5   643 ; Outputs:
                                                01C5   644 ;
                                                01C5   645 ;       Control returns to the requesting process.
                                                01C5   646 ;
                                                01C5   647 ;       The routine destroys R0-R1.
                                                01C5   648 ;
                                                01C5   649 ;--
                                                01C5   650
                                                01C5   651 EXE$ALTQUEPKT::                              ; Start I/O in driver.
                                                01C5   652          DSBINT  UCB$B_FIPL(R5)              ; Raise to fork IPL.
                50    24 A5    D0              01CC   653          MOVL    UCB$L_DDB(R5),R0            ; Get address of unit's DDB.
                50    0C A0    D0              01D0   654          MOVL    DDB$L_DDT(R0),R0            ; Get address of unit's DDT.
DS5.4           51    1C A0    D0              01D4    .1         MOVL    DDT$L_ALTSTART(R0),R1       ; Get start offset
DS5.4                  09       13              01D8    .2         BEQL    20$                         ; Branch if no alternate entry point
DS5.4           03 51    10     E0              01DA    .3         BBS     #16,R1,10$                  ; Branch if supervisor absolute
DS5.4              51    50     C0              01DE    .4         ADDL    R0,R1                       ; Make relative to DDT
-5                                              01E1   660                                             ; address.
                                                01E1   661
                                                01E1   662 10$:
                       61       16              01E1   663          JSB     (R1)                        ; Go to the driver entry point.
                                                01E3   664
                                                01E3   665 20$:
                                                01E3   666          ENBINT                              ; Reenable interrupts.
                                05              01E6   667          RSB                                 ; Return to caller.
```

```
                         01E7   669          .SBTTL  QUEUE I/O PACKET TO ACP
                         01E7   670 ;+
                         01E7   671 ; EXE$QIOACPPKT - QUEUE I/O PACKET TO ACP
                         01E7   672 ;
                         01E7   673 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                         01E7   674 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE ACP.
                         01E7   675 ;
                         01E7   676 ; INPUTS:
                         01E7   677 ;
                         01E7   678 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                         01E7   679 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                         01E7   680 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                         01E7   681 ;
                         01E7   682 ;       CURRENT IPL MUST BE AT SYNCH OR HIGHER LEVEL.
                         01E7   683 ;
                         01E7   684 ; OUTPUTS:
                         01E7   685 ;
                         01E7   686 ;       R4 ALTERED
                         01E7   687 ;       THE I/O PACKET IS QUEUED AT THE END OF THE APPROPRIATE ACP QUEUE
                         01E7   688 ;       AND A NORMAL COMPLETION STATUS IS RETURNED.
                         01E7   689 ;-
                         01E7   690
                         01E7   691 EXE$QIOACPPKT::                          ;QUEUE I/O PACKET TO ACP
-7                       01E7   699          BUG_CHECK NONEXSTACP            ;NONEXISTENT ACP PROCESS
                         01F9   700 EXE$QIORETURN::
                         01F9   701          MOVZWL  #SS$_NORMAL,R0          ;QUEUE I/O REQUEST COMPLETION STATUS RETURN
                   50    0000'8F   3C  01F9   702          MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION STATUS
                         01FE   703 QIORETURN:                               ;RETURN SPECIFIED STATUS
                         01FE   704          SETIPL  #0                      ;ALLOW ALL INTERRUPTS
                               04  0201   705          RET                             ;
```

```
                                0202   706          .SBTTL  INSERT I/O PACKET IN UNIT QUEUE
                                0202   707 ;+
                                0202   708 ; EXE$INSIOQ - INSERT I/O PACKET IN UNIT QUEUE
                                0202   709 ;
                                0202   710 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A UNIT QUEUE AND CALL
                                0202   711 ; THE APPROPRIATE I/O DRIVER IF THE UNIT IS NOT BUSY.
                                0202   712 ;
                                0202   713 ; INPUTS:
                                0202   714 ;
                                0202   715 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                0202   716 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0202   717 ;-
                                0202   718
                                0202   719 EXE$INSIOQ::                                ;INSERT IN I/O QUEUE
                                0202   720          DSBINT  UCB$B_FIPL(R5)             ;RAISE IPL TO FORK LEVEL
05 58 A5    08    E2            0209   721          BBSS    #UCB$V_BSY,UCB$W_STS(R5),10$ ;IF SET, THEN DEVICE IS BUSY
            FDEF'  30           020E   722          BSBW    IOC$INITIATE               ;INITIATE I/O FUNCTION
                06  11          0211   723          BRB     20$                        ;
   52   40 A5      DE           0213   724 10$:     MOVAL   UCB$L_IOQFL(R5),R2         ;GET ADDRESS OF I/O QUEUE LISTHEAD
                04  10          0217   725          BSBB    EXE$INSERTIRP              ;INSERT I/O PACKET IN DEVICE QUEUE
                                0219   726 20$:     ENBINT                             ;ENABLE INTERRUPTS
                            05  021C   727          RSB                                ;
```

```
                     021D   729          .SBTTL  INSERT I/O PACKET IN QUEUE BY PRIORITY
                     021D   730 ;+
                     021D   731 ; EXE$INSERTIRP - INSERT I/O PACKET IN QUEUE BY PRIORITY
                     021D   732 ;
                     021D   733 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A SPECIFIED QUEUE BY
                     021D   734 ; PRIORITY.
                     021D   735 ;
                     021D   736 ; INPUTS:
                     021D   737 ;
                     021D   738 ;       R2 = ADDRESS OF QUEUE LISTHEAD.
                     021D   739 ;       R3 = ADDRESS OF I/O PACKET.
                     021D   740 ;
                     021D   741 ;       CURRENT IPL MUST BE THE FORK LEVEL OF THE RESPECTIVE DRIVER PROCESS
                     021D   742 ;       OR HIGHER.
                     021D   743 ;
                     021D   744 ; OUTPUTS:
                     021D   745 ;
                     021D   746 ;       THE I/O PACKET IS INSERTED IN THE SPECIFIED QUEUE BY PRIORITY AND
                     021D   747 ;       THE 'Z' CONDITION CODE IS RETURNED TO THE CALLER.
                     021D   748 ;
                     021D   749 ;               'Z' = 1 = ENTRY WAS FIRST ENTRY IN THE QUEUE.
                     021D   750 ;
                     021D   751 ;               'Z' = 0 = ENTRIES WERE ALREADY IN THE QUEUE.
                     021D   752 ;
                     021D   753 ;       R2 AND R3 ARE PRESERVED ACROSS THE CALL.
                     021D   754 ;-
                     021D   755
                     021D   756 EXE$INSERTIRP::                              ;INSERT I/O PACKET IN QUEUE BY PRIORITY
          51   52    D0   021D   757          MOVL    R2,R1                   ;COPY LISTHEAD ADDRESS
     51   04 A1    D0   0220   758 10$:      MOVL    IRP$L_IOQBL(R1),R1      ;GET ADDRESS OF NEXT ENTRY
          51   52    D1   0224   759          CMPL    R2,R1                   ;END OF QUEUE?
               07    13   0227   760          BEQL    20$                     ;IF EQL YES
23 A1     23 A3    91   0229   761          CMPB    IRP$B_PRI(R3),IRP$B_PRI(R1) ;NEW ENTRY PRIORITY GREATER?
               F0    1F   022E   762          BLSSU   10$                     ;IF LSS YES
          61   63    0E   0230   763 20$:     INSQUE  IRP$L_IOQFL(R3),IRP$L_IOQFL(R1) ;INSERT PACKET IN I/O QUEUE
                     05   0233   764          RSB                             ;
                          0234   765
                          0234   766          .END
```

```
$ER                     =  00000001     D
$MODULE                    00000000 R   D   02
ACB$B_RMOD                 0000000B     D
ACB$B_TYPE                 0000000A     D
ACB$C_LENGTH               00000018     D
ACB$K_LENGTH               00000018     D
ACB$L_AST                  00000010     D
ACB$L_ASTPRM               00000014     D
ACB$L_ASTQBL               00000004     D
ACB$L_ASTQFL               00000000     D
ACB$L_KAST                 00000018     D
ACB$L_PID                  0000000C     D
ACB$M_QUOTA             =  00000040     D
ACB$V_QUOTA             =  00000006     D
ACB$W_SIZE                 00000008     D
ACCESS                     0000010C R   D   02
ALLOC                      000000B4 R   D   02
ASTADR                  =  00000014     D
ASTPRM                  =  00000018     D
BUG$CHECK                  ********   X   02
CCB$B_AMOD                 00000009     D
CCB$B_STS                  00000008     D
CCB$C_LENGTH               00000010     D
CCB$K_LENGTH               00000010     D
CCB$L_DIRP                 0000000C     D
CCB$L_UCB                  00000000     D
CCB$L_WIND                 00000004     D
CCB$W_IOC                  0000000A     D
CHAN                    =  00000008     D
DDB$B_ACPCLASS             00000013     D
DDB$B_TYPE                 0000000A     D
DDB$C_LENGTH               00000034     D
DDB$K_LENGTH               00000034     D
DDB$L_ACPD                 00000010     D
DDB$L_DDT                  0000000C     D
DDB$L_LINK                 00000000     D
DDB$L_UCB                  00000004     D
DDB$T_DRVNAME              00000024     D
DDB$T_NAME                 00000014     D
DDB$W_SIZE                 00000008     D
DDT$L_ALTSTART             0000001C     D
DDT$L_CANCEL               0000000C     D
DDT$L_FDT                  00000008     D
DDT$L_REGDUMP              00000010     D
DDT$L_START                00000000     D
DDT$L_UNITINIT             00000018     D
DDT$L_UNSOLINT             00000004     D
DDT$W_DIAGBUF              00000014     D
DDT$W_ERRORBUF             00000016     D
DEV$V_FOD                  ********   X   02
DEV$V_FOR                  ********   X   02
DEV$V_MNT                  ********   X   02
DEV$V_SPL                  ********   X   02
DS$AX_SOFTPCB              ********   X   02
DYN$C_IRP               =  0000000A     D
EFN                     =  00000004     D
ERROR                      000000A1 R   D   02
EXE$ABORTIO                0000019A RG  D   02
EXE$ALLOCBUF               ********     X   02
EXE$ALLOCIRP               ********     X   02
EXE$ALTQUEPKT              000001C5 RG  D   02
EXE$FINISHIO               000001A9 RG  D   02
EXE$FINISHIOC              000001A7 RG  D   02
EXE$INSERTIRP              0000021D RG  D   02
EXE$INSIOQ                 00000202 RG  D   02
EXE$QIOACPPKT              000001E7 RG  D   02
EXE$QIODRVPKT              000001C1 RG  D   02
EXE$QIORETURN              000001F9 RG  D   02
FDTACT                  =  00000010     D
FUNC                    =  0000000C     D
GTPKT                      000000BF R   D   02
IO$M_FCODE                 ********     X   02
IO$_LOGICAL                ********     X   02
IO$_PHYSICAL               ********     X   02
IO$_READLBLK               ********     X   02
IO$_READVBLK               ********     X   02
IO$_WRITEVBLK              ********     X   02
IOC$GL_IRPFL               ********     X   02
IOC$GL_PSBL                ********     X   02
IOC$INITIATE               ********     X   02
IOC$VERIFYCHAN             ********     X   02
IOSB                    =  00000010     D
IOTYPE                  =  00000008     D
IPL$_IOPOST             =  00000004     D
IRP$B_CARCON               00000038     D
IRP$B_EFN                  00000022     D
IRP$B_PRI                  00000023     D
IRP$B_RMOD                 0000000B     D
IRP$B_TYPE                 0000000A     D
IRP$C_LENGTH               0000005C     D
IRP$K_LENGTH               0000005C     D
IRP$L_ARB                  00000050     D
IRP$L_AST                  00000010     D
IRP$L_ASTPRM               00000014     D
IRP$L_DIAGBUF              00000044     D
IRP$L_EXTEND               0000004C     D
IRP$L_IOQBL                00000004     D
IRP$L_IOQFL                00000000     D
IRP$L_IOSB                 00000024     D
IRP$L_IOST1                00000034     D
IRP$L_IOST2                00000038     D
IRP$L_MEDIA                00000034     D
IRP$L_PID                  0000000C     D
IRP$L_SEGVBN               00000040     D
IRP$L_SEQNUM               00000048     D
IRP$L_SVAPTE               0000002C     D
IRP$L_TT_TERM              00000038     D
IRP$L_UCB                  0000001C     D
IRP$L_WIND                 00000018     D
IRP$M_BUFIO             =  00000001     D
IRP$M_DIAGBUF           =  00000080     D
IRP$M_PHYSIO            =  00000100     D
IRP$Q_NT_PRVMSK            0000003C     D
IRP$W_ABCNT                0000003C     D
```

```
IRP$W_BCNT                      00000032    D
IRP$W_BOFF                      00000030    D
IRP$W_CHAN                      00000028    D
IRP$W_FUNC                      00000020    D
IRP$W_OBCNT                     0000003E    D
IRP$W_SIZE                      00000008    D
IRP$W_STS                       0000002A    D
IRP$W_TT_PRMPT                  0000003C    D
LEGAL                         = 00000000    D
P1                            = 0000001C    D
P2                            = 00000020    D
P3                            = 00000024    D
P4                            = 00000028    D
P5                            = 0000002C    D
P6                            = 00000030    D
PCB$B_ASTACT                    0000000C    D
PCB$B_ASTEN                     0000000D    D
PCB$B_PRI                       0000000B    D
PCB$B_PRIB                      0000002F    D
PCB$B_TYPE                      0000000A    D
PCB$B_WEFC                      0000002E    D
PCB$C_LENGTH                    0000008C    D
PCB$K_LENGTH                    0000008C    D
PCB$L_ARB                       00000084    D
PCB$L_ASTQBL                    00000014    D
PCB$L_ASTQFL                    00000010    D
PCB$L_EFC2P                     00000058    D
PCB$L_EFC3P                     0000005C    D
PCB$L_EFCS                      00000050    D
PCB$L_EFCU                      00000054    D
PCB$L_EFWM                      0000004C    D
PCB$L_JIB                       00000078    D
PCB$L_OWNER                     0000001C    D
PCB$L_PHD                       00000064    D
PCB$L_PHYPCB                    00000018    D
PCB$L_PID                       00000060    D
PCB$L_PQB                       0000004C    D
PCB$L_SQBL                      00000004    D
PCB$L_SQFL                      00000000    D
PCB$L_STS                       00000024    D
PCB$L_UIC                       00000088    D
PCB$L_WSSWP                     00000020    D
PCB$L_WTIME                     00000028    D
PCB$Q_PRIV                      0000007C    D
PCB$T_LNAME                     00000068    D
PCB$T_TERMINAL                  00000044    D
PCB$W_APTCNT                    00000030    D
PCB$W_ASTCNT                    00000038    D
PCB$W_BIOCNT                    0000003A    D
PCB$W_BIOLM                     0000003C    D
PCB$W_DIOCNT                    0000003E    D
PCB$W_DIOLM                     00000040    D
PCB$W_GPGCNT                    00000034    D
PCB$W_GRP                       0000008A    D
PCB$W_MEM                       00000088    D
PCB$W_MTXCNT                    0000000E    D
PCB$W_PPGCNT                    00000036    D
PCB$W_PRCCNT                    00000042    D
PCB$W_SIZE                      00000008    D
PCB$W_STATE                     0000002C    D
PCB$W_TMBU                      00000032    D
PR$_IPL                       = 00000012    D
PR$_SIRR                      = 00000014    D
PSL$S_PRVMOD                  = 00000002    D
PSL$V_PRVMOD                  = 00000016    D
QIORETURN                       000001FE R  D   02
SIZ...                        = 00000001    D
SS$_ACCVIO                      ********    X   02
SS$_DEVOFFLINE                  ********    X   02
SS$_ILLIOFUNC                   ********    X   02
SS$_NORMAL                      ********    X   02
SUCCES                          000000C8 R  D   02
SYS$CLREF                       ********    X   02
SYS$SETEF                       ********    X   02
UCB$B_AMOD                      00000053    D
UCB$B_CEX                       00000077    D
UCB$B_CM1                       0000004A    D
UCB$B_CM2                       0000004B    D
UCB$B_DEVCLASS                  00000038    D
UCB$B_DEVTYPE                   00000039    D
UCB$B_DIPL                      00000052    D
UCB$B_DX_SCTCNT                 000000A6    D
UCB$B_ERTCNT                    00000070    D
UCB$B_ERTMAX                    00000071    D
UCB$B_FEX                       00000076    D
UCB$B_FIPL                      0000000B    D
UCB$B_LOCSRV                    0000003C    D
UCB$B_OFFNDX                    00000094    D
UCB$B_OFFRTC                    00000095    D
UCB$B_REMSRV                    0000003D    D
UCB$B_SECTORS                   0000003C    D
UCB$B_SLAVE                     00000074    D
UCB$B_SPR                       00000075    D
UCB$B_STATE                     00000052    D
UCB$B_TRACKS                    0000003D    D
UCB$B_TT_CRFILL                 0000009D    D
UCB$B_TT_DECRF                  000000A1    D
UCB$B_TT_DELFF                  000000A2    D
UCB$B_TT_DESPEE                 000000A0    D
UCB$B_TT_DETYPE                 000000A4    D
UCB$B_TT_LFFILL                 0000009E    D
UCB$B_TT_SPEED                  0000009C    D
UCB$B_TYPE                      0000000A    D
UCB$B_VERTSZ                    0000003F    D
UCB$C_LENGTH                    00000074    D
UCB$C_MB_LENGTH                 00000090    D
UCB$C_TT_LENGTH                 000000BC    D
UCB$K_LENGTH                    00000074    D
UCB$K_MB_LENGTH                 00000090    D
UCB$K_TT_LENGTH                 000000BC    D
UCB$L_AMB                       00000054    D
UCB$L_ASTQBL                    00000010    D
UCB$L_ASTQFL                    0000000C    D
UCB$L_CPID                      0000005C    D
```

UCB$L_CRB		00000020	D	UCB$W_DIRSEQ		00000088	D
UCB$L_DDB		00000024	D	UCB$W_DSTADDR		00000018	D
UCB$L_DEVCHAR		00000034	D	UCB$W_DX_BCR		000000A4	D
UCB$L_DEVDEPEND		0000003C	D	UCB$W_ECT		00000090	D
UCB$L_DPC		00000080	D	UCB$W_EC2		00000092	D
UCB$L_DUETIM		0000005C	D	UCB$W_ERRCNT		00000072	D
UCB$L_DX_BFPNT		0000009C	D	UCB$W_FUNC		0000007E	D
UCB$L_DX_BUF		00000098	D	UCB$W_MB_SEED		00000000	D
UCB$L_DX_RXDB		000000A0	D	UCB$W_MSGCNT		00000016	D
UCB$L_EMB		00000078	D	UCB$W_MSGMAX		00000014	D
UCB$L_FIRST		00000014	D	UCB$W_NT_CHAN		0000007C	D
UCB$L_FPC		0000000C	D	UCB$W_OFFSET		0000008A	D
UCB$L_FQBL		00000004	D	UCB$W_REFC		00000050	D
UCB$L_FQFL		00000000	D	UCB$W_SIZE		00000008	D
UCB$L_FR3		00000010	D	UCB$W_SRCADDR		0000001A	D
UCB$L_FR4		00000014	D	UCB$W_STS		00000058	D
UCB$L_IOQBL		00000044	D	UCB$W_TT_DESIZE		000000A5	D
UCB$L_IOQFL		00000040	D	UCB$W_UNIT		00000048	D
UCB$L_IRP		0000004C	D	UCB$W_VPROT		0000001A	D
UCB$L_LINK		0000002C	D	VCB$B_BLOCKFACT		00000052	D
UCB$L_LOGADR		00000064	D	VCB$B_CUR_RVN		0000002F	D
UCB$L_MAXBLOCK		00000084	D	VCB$B_EOFDELTA		0000004E	D
UCB$L_MB_MBX		0000007C	D	VCB$B_IBMAPSIZE		00000038	D
UCB$L_MB_PORT		0000008C	D	VCB$B_IBMAPVBN		0000003A	D
UCB$L_MB_RAST		00000078	D	VCB$B_LRU_LIM		00000049	D
UCB$L_MB_SHB		00000080	D	VCB$B_QNAMECNT		0000000B	D
UCB$L_MB_WAST		00000074	D	VCB$B_RESFILES		0000004F	D
UCB$L_MB_WIOQBL		00000088	D	VCB$B_SBMAPSIZE		00000039	D
UCB$L_MB_WIOQFL		00000084	D	VCB$B_SBMAPVBN		00000038	D
UCB$L_MEDIA		0000008C	D	VCB$B_STATUS		0000000B	D
UCB$L_NT_DATSSB		00000074	D	VCB$B_STATUS2		00000053	D
UCB$L_NT_INTSSB		00000078	D	VCB$B_TM		0000002E	D
UCB$L_OPCNT		00000060	D	VCB$B_TYPE		0000000A	D
UCB$L_OWNUIC		0000001C	D	VCB$B_WINDOW		00000048	D
UCB$L_PID		00000028	D	VCB$C_COMLEN		00000024	D
UCB$L_RQBL		00000004	D	VCB$C_LENGTH		00000064	D
UCB$L_RQFL		00000000	D	VCB$C_MRKLEN		0000000B	D
UCB$L_SVAPTE		00000068	D	VCB$K_COMLEN		00000024	D
UCB$L_SVPN		00000064	D	VCB$K_LENGTH		00000064	D
UCB$L_TT_DECHAR		000000A8	D	VCB$K_MRKLEN		0000000B	D
UCB$L_TT_RDUE		0000008C	D	VCB$L_AQB		00000010	D
UCB$L_TT_RTIMOU		000000B8	D	VCB$L_BLOCKBL		00000004	D
UCB$L_VCB		00000030	D	VCB$L_BLOCKFL		00000000	D
UCB$T_PARTNER		0000000C	D	VCB$L_CACHE		00000058	D
UCB$V_BSY	=	00000008	D	VCB$L_CUR_FID		00000024	D
UCB$V_ONLINE	=	00000004	D	VCB$L_FCBBL		00000004	D
UCB$W_BCNT		0000006E	D	VCB$L_FCBFL		00000000	D
UCB$W_BCR		00000096	D	VCB$L_FREE		00000040	D
UCB$W_BOFF		0000006C	D	VCB$L_HOME2LBN		00000028	D
UCB$W_BUFQUO		00000018	D	VCB$L_HOMELBN		00000024	D
UCB$W_BYTESTOGO		0000003E	D	VCB$L_IBMAPLBN		00000030	D
UCB$W_CHARGE		0000004A	D	VCB$L_IXHDR2LBN		0000002C	D
UCB$W_CYLINDERS		0000003E	D	VCB$L_MAXFILES		00000044	D
UCB$W_DA		0000008C	D	VCB$L_MVL		00000034	D
UCB$W_DC		0000008E	D	VCB$L_QUOCACHE		0000005C	D
UCB$W_DEVBUFSIZ		0000003A	D	VCB$L_QUOTAFCB		00000054	D
UCB$W_DEVSTS		0000005A	D	VCB$L_RVT		00000020	D

```
VCB$L_SBMAPLBN                  00000034     D
VCB$L_START_FID                 00000028     D
VCB$L_ST_RECORD                 00000030     D
VCB$L_USRLBLAST                 00000044     D
VCB$L_VPBL                      00000040     D
VCB$L_VPFL                      0000003C     D
VCB$L_WCB                       00000038     D
VCB$T_QNAME                     0000000C     D
VCB$T_VOLNAME                   00000014     D
VCB$W_CLUSTER                   0000003C     D
VCB$W_CUR_NUM                   00000024     D
VCB$W_CUR_SEQ                   00000026     D
VCB$W_EXTEND                    0000003E     D
VCB$W_FILEPROT                  0000004A     D
VCB$W_MCOUNT                    0000C04C     D
VCB$W_MODE                      0000002C     D
VCB$W_PENDERR                   00000062     D
VCB$W_QUOSIZE                   00000060     D
VCB$W_RECORDSZ                  00000050     D
VCB$W_RVN                       0000000E     D
VCB$W_SIZE                      00000008     D
VCB$W_START_NUM                 00000028     D
VCB$W_START_SEQ                 0000002A     D
VCB$W_TRANS                     0000000C     D
VMS$QIO                         00000007 RG  D  02
WCB$B_ACCESS                    0000000B     D
WCB$B_TYPE                      0000000A     D
WCB$C_LENGTH                    00000024     D
WCB$C_MAP                       00000024     D
WCB$K_LENGTH                    00000024     D
WCB$K_MAP                       00000024     D
WCB$L_FCB                       00000018     D
WCB$L_LBN                       00000002     D
WCB$L_ORGUCB                    00000010     D
WCB$L_P1_LBN                    00000026     D
WCB$L_P2_LBN                    0000002C     D
WCB$L_PID                       0000000C     D
WCB$L_PREVLBN                   FFFFFFFC     D
WCB$L_RVT                       0000001C     D
WCB$L_STVBN                     00000020     D
WCB$L_WLBL                      00000004     D
WCB$L_WLFL                      00000000     D
WCB$W_ACON                      00000014     D
WCB$W_COUNT                     00000000     D
WCB$W_NMAP                      00000016     D
WCB$W_P1_COUNT                  00000024     D
WCB$W_P2_COUNT                  0000002A     D
WCB$W_PREVCOUNT                 FFFFFFFA     D
WCB$W_REFCNT                    0000000E     D
WCB$W_SIZE                      00000008     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000234 (564.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	119.2 (1)	
$MODULE	00000000-R	119.2 (1)	
ACB$M_QUOTA	=00000040		#-398 (1)
ACB$V_QUOTA	=00000006		#-585 (1)
ACCESS	0000010C-R	305 (1)	
ALLOC	000000B4-R	255.1 (1)	#-260 (1)
ASTADR	=00000014	104 (1)	#-276 (1)
ASTPRM	=00000018	105 (1)	
BUG$CHECK	00000000-XR		699 (1)
CCB$L_UCB	00000000		#-203 (1)
CCB$L_WIND	00000004		#-229 (1) #-276.1 (1)
CCB$W_IOC	0000000A		#-268 (1)
CHAN	=00000008	101 (1)	#-199 (1)
DDB$L_DDT	0000000C		#-212 (1) #-376 (1) #-654 (1)
DDT$L_ALTSTART	0000001C		#-654.1 (1)
DDT$L_FDT	00000008		#-212.1 (1)
DDT$W_DIAGBUF	00000014		#-377 (1)
DEV$V_FOD	00000000-XR		#-309 (1)
DEV$V_FOR	00000000-XR		#-318 (1)
DEV$V_MNT	00000000-XR		#-317 (1)
DEV$V_SPL	00000000-XR		#-207 (1)
DS$AX_SOFTPCB	00000000-XR		195.2 (1)
DYN$C_IRP	=0000000A		#-272 (1)
EFN	=00000004	100 (1)	#-195.3 (1) #-237.1 (1) #-286 (1)
ERROR	000000A1-R	236 (1)	#-195.6 (1) #-202 (1) #-232 (1) #-234 (1)
			#-258 (1)
EXE$ABORTIO	0000019A-R	583 (1)	#-363 (1)
EXE$ALLOCBUF	00000000-XR		379.1 (1)
EXE$ALLOCIRP	00000000-XR		255.1 (1)
EXE$ALTQUEPKT	000001C5-R	651 (1)	
EXE$FINISHIO	000001A9-R	590 (1)	
EXE$FINISHIOC	000001A7-R	588 (1)	
EXE$INSERTIRP	0000021D-R	756 (1)	#-725 (1)
EXE$INSIOQ	00000202-R	719 (1)	#-620 (1)
EXE$QIOACPPKT	000001E7-R	691 (1)	
EXE$QIODRVPKT	000001C1-R	619 (1)	
EXE$QIORETURN	000001F9-R	700 (1)	#-621 (1)
FDTACT	=00000010	119 (1)	#-418 (1)
FUNC	=0000000C	102 (1)	#-205 (1)
GTPKT	000000BF-R	258.1 (1)	#-229 (1)
IO$M_FCODE	00000000-XR		#-206 (1)
IO$_LOGICAL	00000000-XR		#-208 (1)
IO$_PHYSICAL	00000000-XR		#-369 (1)
IO$_READLBLK	00000000-XR		#-324 (1) #-325 (1)
IO$_READVBLK	00000000-XR		#-307 (1) #-324 (1) #-325 (1)
IO$_WRITEVBLK	00000000-XR		#-305 (1)
IOC$GL_IRPFL	00000000-XR		258.1 (1)
IOC$GL_PSBL	00000000-XR		593.1 (1)
IOC$INITIATE	00000000-XR		#-722 (1)
IOC$VERIFYCHAN	00000000-XR		#-200 (1)

```
IOSB                    =00000010     103  (1)    #-218    (1)    #-288    (1)
IOTYPE                  =00000008     118  (1)      224    (1)      290    (1)
IPL$_IOPOST             =00000004                  #-596    (1)
IRP$B_PRI                00000023                  #-761    (1)
IRP$B_RMOD               0000000B                  #-398    (1)      585    (1)
IRP$C_LENGTH             0000005C                  #-271    (1)
IRP$L_AST                00000010                  #-392    (1)
IRP$L_DIAGBUF            00000044                  #-383    (1)
IRP$L_IOQBL              00000004                  #-758    (1)
IRP$L_IOQFL              00000000                    763    (1)
IRP$L_IOSB               00000024                  #-584    (1)
IRP$L_MEDIA              00000034                  #-592    (1)
IRP$L_WIND               00000018                  #-315    (1)
IRP$M_BUFIO             =00000001                  #-291    (1)
IRP$M_DIAGBUF           =00000080                  #-386    (1)
IRP$M_PHYSIO            =00000100                  #-371    (1)
IRP$W_FUNC               00C00020                  #-325    (1)
IRP$W_SIZE               00000008                  #-270    (1)
IRP$W_STS                0000002A                  #-371    (1)    #-386    (1)
LEGAL                   =00000000     117  (1)      216    (1)
P1                      =0000001C     106  (1)    #-419    (1)
P2                      =00000020     107  (1)
P3                      =00000024     108  (1)
P4                      =00000028     109  (1)
P5                      =0000002C     110  (1)
P6                      =00000030     111  (1)    #-372    (1)
PCB$L_PID                00000060                  #-275    (1)
PCB$W_ASTCNT             00000038                  #-586    (1)
PCB$W_BIOCNT             0000003A                    223    (1)
PCB$W_DIOCNT             0000003E                    225    (1)
PR$_IPL                 =00000012                  #-236    (1)    #-652    (1)    #-666    (1)    #-703    (1)
                                                   #-720    (1)    #-726    (1)
PR$_SIRR                =00000014                  #-596    (1)
PSL$S_PRVMOD            =00000002                  #-273    (1)
PSL$V_PRVMOD            =00000016                  #-273    (1)
QIORETURN                000001FE-R    702  (1)    #-595    (1)    #-597    (1)
SS$_ACCVIO               00000000-XR               #-235    (1)
SS$_DEVOFFLINE           00000000-XR               #-233    (1)
SS$_ILLIOFUNC            00000000-XR               #-231    (1)
SS$_NORMAL               00000000-XR               #-593    (1)    #-701    (1)
SUCCES                   000000C8-R    267  (1)    #-257    (1)
SYS$CLREF                00000000-XR                 195.4  (1)
SYS$SETEF                00000000-XR                 237.2  (1)
UCB$B_FIPL               0000000B                  #-652    (1)    #-720    (1)
UCB$L_AMB                00000054                  #-210    (1)
UCB$L_DDB                00000024                  #-211    (1)    #-375    (1)    #-653    (1)
UCB$L_DEVCHAR            00000034                    207    (1)      309    (1)      317    (1)      318    (1)
UCB$L_IOQFL              00000040                    224    (1)
UCB$L_OPCNT              00000060                  #-591    (1)
UCB$V_BSY               =00000008                  #-721    (1)
UCB$V_ONLINE            =00000004                  #-217    (1)
UCB$W_STS                00000058                    217    (1)      721    (1)
VMS$QIO                  00000007-R    195.1  (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	73 (1)	73 (1)
$AQBDEF	2	74 (1)	74 (1)
$CADEF	1	75 (1)	75 (1)
$CCBDEF	1	76 (1)	76 (1)
$DDBDEF	1	77 (1)	77 (1)
$DDTDEF	1	78 (1)	78 (1)
$DEFINI	1	73 (1)	73 (1) 74 (1) 75 (1) 76 (1) 77 (1)
			78 (1) 79 (1) 80 (1) 81 (1) 82 (1)
			83 (1) 84 (1) 85 (1) 86 (1) 87 (1)
			88 (1) 89 (1) 90 (1) 91 (1) 92 (1)
$DYNDEF	2	79 (1)	79 (1)
$IPLDEF	1	80 (1)	80 (1)
$IRPDEF	4	81 (1)	81 (1)
$PCBDEF	4	82 (1)	82 (1)
$PHDDEF	6	83 (1)	83 (1)
$PRDEF	4	84 (1)	84 (1)
$PRIDEF	1	85 (1)	85 (1)
$PRVDEF	4	86 (1)	86 (1)
$PSLDEF	2	87 (1)	87 (1)
$RSNDEF	1	88 (1)	88 (1)
$SECDEF	3	89 (1)	89 (1)
$UCBDEF	10	90 (1)	90 (1)
$VCBDEF	6	91 (1)	91 (1)
$WCBDEF	3	92 (1)	92 (1)
BUG_CHECK	1	699 (1)	699 (1)
DSBINT	1	652 (1)	652 (1) 720 (1)
ENBINT	1	666 (1)	666 (1) 726 (1)
IFNOWRT	1	220 (1)	220 (1)
MODNAM	1	119.2 (1)	119.2 (1)
SETIPL	1	236 (1)	236 (1) 703 (1)
SOFTINT	1	596 (1)	596 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.11	00:00:00.42
Command processing	144	00:00:00.82	00:00:02.53
Pass 1	817	00:00:19.61	00:00:33.07
Symbol table sort	0	00:00:01.60	00:00:03.59
Pass 2	160	00:00:04.37	00:00:08.10
Symbol table output	49	00:00:00.33	00:00:00.33
Psect synopsis output	6	00:00:00.02	00:00:00.02
Cross-reference output	51	00:00:00.51	00:00:00.75
Assembler run totals	1266	00:00:27.37	00:00:48.88

The working set limit was 1000 pages.
103970 bytes (204 pages) of virtual memory were used to buffer the intermediate code.

There were 60 pages of symbol table space allocated to hold 1046 non-local and 26 local symbols.
592 source lines were read in Pass 1, producing 0 object records in Pass 2.
141 pages of virtual memory were used to define 36 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	2
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	11
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	10
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	32

1523 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) QIOREQ/UPDA=(QIOREQ.UPD,QIOREQ.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 50 INITIALIZE DRIVER DATA BASE
(1) 79 REINITIALIZE DRIVER DATA BASE
(1) 114 LOCAL SUBROUTINE TO EXECUTE OPERATION CODE
(1) 202 LOCAL SUBROUTINE TO LOCATE DATA STRUCTURE TYPE

```
DS5.4                                   0000     .1          .TITLE  RELOCDRV        *** RELOCDRV relocate I/O driver
DS5.4                                   0000     .2          .IDENT  /01/
DS5.4                                00000000    .3          .PSECT  SEP,SHR,EXE,WRT,LONG
DS5.4                                   0000     .4
-2                                      0000      3
                                        0000      4 ;
                                        0000      5 ;*****************************************************************************
                                        0000      6 ;*                                                                           *
DS5.4                                   0000     .1 ; COPYRIGHT (C) 1978, 1980
-1                                      0000      8 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                                        0000      9 ;*                                                                           *
                                        0000     10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  *
                                        0000     11 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                                        0000     12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER  *
                                        0000     13 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
                                        0000     14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  *
                                        0000     15 ;*  TRANSFERRED.                                                            *
                                        0000     16 ;*                                                                           *
                                        0000     17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  *
                                        0000     18 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                                        0000     19 ;*  CORPORATION.                                                            *
                                        0000     20 ;*                                                                           *
                                        0000     21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  *
                                        0000     22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
                                        0000     23 ;*                                                                           *
                                        0000     24 ;*****************************************************************************
                                        0000     25 ;
                                        0000     26 ; D. N. CUTLER 16-JUN-78
DS5.4                                   0000     .1 ;
DS5.4                                   0000     .2 ;       DS5.4   John Ciukaj     5-May-1980      Version 5.4
DS5.4                                   0000     .3 ;                 Used Vax/Vms version 2.0 RELOCDRV.MAR and created
DS5.4                                   0000     .4 ;                 a UPD file with the edits necessary to reflect the
DS5.4                                   0000     .5 ;                 changes exhibited in Supervisor version 5.3 module
DS5.4                                   0000     .6 ;                 RELOCDRV.
DS5.4                                   0000     .7 ;
                                        0000     27 ;
                                        0000     28 ; RELOCATE DRIVER DATABASE
                                        0000     29 ;
                                        0000     30 ; MACRO LIBRARY CALLS
                                        0000     31 ;
                                        0000     32
                                        0000     33          $CRBDEF                             ;DEFINE CRB OFFSETS
                                        0000     34          $DDBDEF                             ;DEFINE DDB OFFSETS
                                        0000     35          $DPTDEF                             ;DEFINE DPT OFFSETS
                                        0000     36          $DYNDEF                             ;DEFINE DATA STRUCTURE TYPES
                                        0000     37          $IDBDEF                             ;DEFINE IDB OFFSETS
                                        0000     38          $UCBDEF                             ;DEFINE UCB OFFSETS
                                        0000     39          $VECDEF                             ;DEFINE VEC OFFSETS
                                        0000     40
                                        0000     41 ;
                                        0000     42 ; LOCAL DATA
                                        0000     43 ;
                                        0000     44 ; STRUCTURE TYPE BYTE ARRAY
                                        0000     45 ;
                                        0000     46
-1                          10 09 06 05 0000     48 STYPE:  .BYTE   DYN$C_CRB,DYN$C_DDB,DYN$C_IDB,DYN$C_UCB ;
```

```
                        0C04     50          .SBTTL  INITIALIZE DRIVER DATA BASE
                        0004     51 ;+
                        0004     52 ; IOC$INITDRV - INITIALIZE DRIVER DATA BASE
                        0004     53 ;
                        0004     54 ; THIS ROUTINE IS CALLED AFTER HAVING LOADED ALL OR PART OF A DEVICE DATA BASE.
                        0004     55 ; IT MUST BE CALLED FOR EACH UNIT THAT IS DEFINED. ITS FUNCTION IS TO USE THE
                        0004     56 ; INITIALIZATION TABLE IN THE DRIVER PROLOGUE TO INITIALIZE STATIC INFORMATION
                        0004     57 ; IN THE DRIVER DATA BASE.
                        0004     58 ;
                        0004     59 ; INPUTS:
                        0004     60 ;
                        0004     61 ;       R4 = ADDRESS OF DRIVER PROLOGUE TABLE.
                        0004     62 ;       R5 = ADDRESS OF DEVICE UCB.
                        0004     63 ;
                        C004     64 ; OUTPUTS:
                        0004     65 ;
                        0004     66 ;       THE DRIVER PROLOGUE TABLE IS INTERPRETED AND THE RESPECTIVE DATA BASE
                        0004     67 ;       IS INITIALIZED.
                        0004     68 ;
                        0004     69 ;       R0 LOW BIT CLEAR INDICATES A BAD DATA STRUCTURE.
                        0004     70 ;
                        0004     71 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                        0004     72 ;-
                        0004     73
                        0004     74          .ENABL  LSB
                        0004     75 IOC$INITDRV::                                ;INITIALIZE DRIVER DATA BASE
          53   10 A4    3C  0004     76          MOVZWL  DPT$W_INITTAB(R4),R3    ;GET OFFSET TO INITIALIZATION TABLE
               04    11    0008     77          BRB     10$                     ;BR TO COMMON CODE
```

```
                                000A     79          .SBTTL  REINITIALIZE DRIVER DATA BASE
                                000A     80 ;+
                                000A     81 ; IOC$REINITDRV - REINITIALIZE DRIVER DATA BASE
                                000A     82 ;
                                000A     83 ; THIS ROUTINE IS CALLED AFTER HAVING LOADED A DRIVER WHOSE DEVICE DATA BASE IS
                                000A     84 ; ALREADY RESIDENT IN MEMORY. IT MUST BE CALLED FOR EACH UNIT THAT IS DEFINED.
                                000A     85 ; ITS FUNCTION IS TO INITIALIZE ONLY THOSE FIELDS IN THE DATA BASE THAT ARE NOT
                                000A     86 ; DYNAMIC.
                                000A     87 ;
                                000A     88 ; INPUTS:
                                000A     89 ;
                                000A     90 ;       R4 = ADDRESS OF DRIVER PROLOGUE TABLE.
                                000A     91 ;       R5 = ADDRESS OF DEVICE UCB.
                                000A     92 ;
                                C00A     93 ; OUTPUTS:
                                000A     94 ;
                                000A     95 ;       THE DRIVER PROLOGUE TABLE IS INTERPRETED AND THE RESPECTIVE DATA BASE
                                000A     96 ;       IS REINITIALIZED.
                                000A     97 ;
                                000A     98 ;       R0 LOW BIT CLEAR INDICATES A BAD DATA STRUCTURE.
                                000A     99 ;
                                000A    100 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                000A    101 ;-
                                000A    102
                                000A    103 IOC$REINITDRV::                             ;REINITIALIZE DRIVER DATA BASE
          53    12 A4    3C     000A    104         MOVZWL  DPT$W_REINITTAB(R4),R3      ;GET OFFSET TO REINITIALIZATION TABLE
             53    54    C0     000E    105         ADDL    R4,R3                       ;CALCULATE ADDRESS OF INITIALIZE TABLE
                   5F    10     0011    106 20$:    BSBB    LOCATE                      ;LOCATE STRUCTURE TYPE
          51       83    9A     0013    107         MOVZBL  (R3)+,R1                    ;GET OFFSET IN STRUCTURE
          52       51    C0     0016    108         ADDL    R1,R2                       ;CALCULATE BASE ADDRESS OF FIELD
          51       83    9A     0019    109         MOVZBL  (R3)+,R1                    ;GET OPERATION CODE
                   02    10     001C    110         BSBB    EXECUTE                     ;EXECUTE OPERATION CODE
                   F1    11     001E    111         BRB     20$                         ;
                               0020    112         .DSABL  LSB
```

```
                     0020   114          .SBTTL  LOCAL SUBROUTINE TO EXECUTE OPERATION CODE
                     0020   115 ;
                     0020   116 ; EXECUTE - EXECUTE OPERATION CODE
                     0020   117 ;
                     0020   118 ; THIS ROUTINE IS CALLED TO EXECUTE A SINGLE RELOCATION OPERATION CODE.
                     0020   119 ;
                     0020   120 ; INPUTS:
                     0020   121 ;
                     0020   122 ;       R1 = OPERATION CODE.
                     0020   123 ;       R2 = STRUCTURE FIELD BASE ADDRESS.
                     0020   124 ;       R3 = ADDRESS OF NEXT RELOCATION BYTE.
                     0020   125 ;       R4 = BASE ADDRESS OF DRIVER PROLOGUE TABLE.
                     0020   126 ;       R5 = ADDRESS OF DEVICE UCB.
                     0020   127 ;
                     0020   128 ; OUTPUTS:
                     0020   129 ;
                     0020   130 ;       IF AN INVALID OPERATION CODE IS SPECIFIED, A FAILURE IS RETURNED TO THE
                     0020   131 ;       CALLERS CALLER. ELSE THE OPERAND VALUE IS STORE IN THE SPECIFIED FIELD
                     0020   132 ;       IN THE SPECIFIED STRUCTURE.
                     0020   133 ;
                     0020   134
                     0020   135 EXECUTE:                                      ;EXECUTE RELOCATION OPERATION
                     0020   136          CASE    R1,<-                       ;DISPATCH TO FETCH ROUTINE
                     0020   137                  10$,-                       ;BYTE IMMEDIATE
                     0020   138                  30$,-                       ;WORD IMMEDIATE
                     0020   139                  50$,-                       ;WORD DISPLACED
                     0020   140                  70$,-                       ;LONGWORD IMMEDIATE
                     0020   141                  90$,-                       ;FIELD IMMEDIATE
                     0020   142                  >
                     002E   143
                     002E   144 ;
                     002E   145 ; INDIRECT OPERAND
                     002E   146 ;
                     002E   147
50   93   D0         002E   148          MOVL    @(R3)+,R0                   ;GET DATA LONGWORD
                     0031   149          CASE    R1,<-                       ;DISPATCH TO STORE ROUTINE
                     0031   150                  20$,-                       ;STORE BYTE
                     0031   151                  40$,-                       ;STORE WORD
                     0031   152                  60$,-                       ;STORE WORD DISPLACED
                     0031   153                  80$,-                       ;STORE LONGWORD
                     0031   154                  100$,-                      ;STORE FILED
                     0031   155                  >,LIMIT=#^X80               ;
     8E   D5         0041   156          TSTL    (SP)+                       ;REMOVE RETURN FROM STACK
     50   D4         0043   157          CLRL    R0                          ;SET FAILURE INDICATION
          05         0045   158          RSB                                 ;
                     0046   159
                     0046   160 ;
                     0046   161 ; BYTE IMMEDIATE
                     0046   162 ;
                     0046   163
50   83   9A         0046   164 10$:     MOVZBL  (R3)+,R0                    ;GET DATA BYTE
62   50   90         0049   165 20$:     MOVB    R0,(R2)                     ;STORE BYTE
          05         004C   166          RSB                                 ;
                     004D   167
                     004D   168 ;
                     004D   169 ; WORD IMMEDIATE
                     004D   170 ;
```

```
                        004D    171
 50   83   3C   004D    172 30$:    MOVZWL  (R3)+,R0                ;GET DATA WORD
 62   50   B0   0050    173 40$:    MOVW    R0,(R2)                 ;STORE WORD
             05 0053    174         RSB                             ;
                0054    175
                0054    176 ;
                0054    177 ; WORD DISPLACED
                0054    178 ;
                0054    179
 50   83   32   0054    180 50$:    CVTWL   (R3)+,R0                ;GET WORD DISPLACEMENT
 50   54   C0   0057    181 60$:    ADDL    R4,R0                   ;CALCULATE ACTUAL ADDRESS
           03   11 005A  182         BRB     80$                     ;
                005C    183
                005C    184 ;
                C05C    185 ; LONGWORD IMMEDIATE
                005C    186 ;
                005C    187
 50   83   D0   005C    188 70$:    MOVL    (R3)+,R0                ;GET DATA LONGWORD
 62   50   D0   005F    189 80$:    MOVL    R0,(R2)                 ;STORE DATA LONGWORD
             05 0062    190         RSB                             ;
                0063    191
                0063    192 ;
                0063    193 ; FIELD STRUCTURE
                0063    194 ;
                0063    195
 50   83   D0   0063    196 90$:    MOVL    (R3)+,R0                ;GET DATA LONGWORD
 7E   83   9A   0066    197 100$:   MOVZBL  (R3)+,-(SP)             ;GET STARTING BIT NUMBER
 51   83   9A   0069    198         MOVZBL  (R3)+,R1                ;GET SIZE OF FIELD
51    8E   50   F0 006C  199         INSV    R0,(SP)+,R1,(R2)        ;STORE FIELD
           62           0070
             05 0071    200         RSB                             ;
```

```
                        0072   202          .SBTTL  LOCAL SUBROUTINE TO LOCATE DATA STRUCTURE TYPE
                        0072   203 ;
                        0072   204 ; LOCATE - LOCATE DATA STRUCTURE TYPE
                        0072   205 ;
                        0072   206 ; INPUTS:
                        0072   207 ;
                        0072   208 ;       R5 = ADDRESS OF DEVICE UCB.
                        0072   209 ;
                        0072   210 ; OUTPUTS:
                        0072   211 ;
                        0072   212 ;       IF THE STRUCTURE TYPE CODE IS ZERO, THEN SUCCESS IS RETURNED TO THE CALLERS
                        0072   213 ;       CALLER. IF THE STRUCTURE TYPE CODE IS INVALID, THEN AN ERROR IS RETURNED
                        0072   214 ;       TO THE CALLERS CALLER. ELSE THE APPROPRIATE STRUCTURE BASE ADDRESS IS
                        0072   215 ;       RETURNED IN R2.
                        0072   216 ;
                        0072   217
                        0072   218 LOCATE:                                              ;LOCATE DATA STRUCTURE TYPE
50    01    88          0072   219          BISB    #1,R0                            ;ASSUME END OF RELOCATION TABLE
      63    95          0075   220          TSTB    (R3)                             ;END OF TABLE?
      11    13          0077   221          BEQL    10$                              ;IF EQL YES
04    83    3A          0079   222          LOCC    (R3)+,#4,STYPE                   ;LOCATE STRUCTURE CODE
   62 AF                007C
                        007E   223          CASE    R0,<-                            ;DISPATCH TO STRUCTURE ROUTINE
                        007E   224                  20$,-                            ;UCB
                        007E   225                  30$,-                            ;IDB
                        007E   226                  40$,-                            ;DDB
                        007E   227                  50$,-                            ;CRB
                        007E   228                  >,LIMIT=#1                       ;
      8E    D5          008A   229 10$:     TSTL    (SP)+                            ;REMOVE RETURN FROM STACK
            05          008C   230          RSB                                      ;
                        008D   231
                        008D   232 ;
                        008D   233 ; UCB STRUCTURE
                        008D   234 ;
                        008D   235
   52   55    D0          008D   236 20$:     MOVL    R5,R2                            ;SET UCB ADDRESS
            05          0090   237          RSB                                      ;
                        0091   238
                        0091   239 ;
                        0091   240 ; IDB STRUCTURE
                        0091   241 ;
                        0091   242
52    20 A5    D0          0091   243 30$:     MOVL    UCB$L_CRB(R5),R2                 ;GET ADDRESS OF CRB
52    1C A2    D0          0095   244          MOVL    CRB$L_INTD+VEC$L_IDB(R2),R2 ;GET ADDRESS OF IDB
            05          0099   245          RSB                                      ;
                        009A   246
                        009A   247 ;
                        009A   248 ; DDB STRUCTURE
                        009A   249 ;
                        009A   250
52    24 A5    D0          009A   251 40$:     MOVL    UCB$L_DDB(R5),R2                 ;GET ADDRESS OF DDB
            05          009E   252          RSB                                      ;
                        009F   253
                        009F   254 ;
                        009F   255 ; CRB STRUCTURE
                        009F   256 ;
                        009F   257
```

```
                52   20 A5   D0  009F   258 50$:    MOVL    UCB$L_CRB(R5),R2        ;GET ADDRESS OF CRB
                             05  00A3   259         RSB                             ;
                                 00A4   260
                                 00A4   261         .END
```

CRB$B_MASK	0000000E	D		UCB$B_AMOD	00000053	D		UCB$L_IOQFL	00000040	D
CRB$B_TYPE	0000000A	D		UCB$B_CEX	00000077	D		UCB$L_IRP	0000004C	D
CRB$C_LENGTH	00000038	D		UCB$B_CM1	0000004A	D		UCB$L_LINK	0000002C	D
CRB$K_LENGTH	00000038	D		UCB$B_CM2	0000004B	D		UCB$L_LOGADR	00000064	D
CRB$L_INTD	00000014	D		UCB$B_DEVCLASS	00000038	D		UCB$L_MAXBLOCK	00000084	D
CRB$L_INTD2	00000038	D		UCB$B_DEVTYPE	00000039	D		UCB$L_MB_MBX	0000007C	D
CRB$L_LINK	00000010	D		UCB$B_DIPL	00000052	D		UCB$L_MB_PORT	0000008C	D
CRB$L_WQBL	00000004	D		UCB$B_DX_SCTCNT	000000A6	D		UCB$L_MB_RAST	00000078	D
CRB$L_WQFL	00000000	D		UCB$B_ERTCNT	00000070	D		UCB$L_MB_SHB	00000080	D
CRB$W_REFC	0000000C	D		UCB$B_ERTMAX	00000071	D		UCB$L_MB_WAST	00000074	D
CRB$W_SIZE	00000008	D		UCB$B_FEX	00000076	D		UCB$L_MB_WIOQBL	00000088	D
DDB$B_ACPCLASS	00000013	D		UCB$B_FIPL	0000000B	D		UCB$L_MB_WIOQFL	00000084	D
DDB$B_TYPE	0000000A	D		UCB$B_LOCSRV	0000003C	D		UCB$L_MEDIA	0000008C	D
DDB$C_LENGTH	00000034	D		UCB$B_OFFNDX	00000094	D		UCB$L_NT_DATSSB	00000074	D
DDB$K_LENGTH	00000034	D		UCB$B_OFFRTC	00000095	D		UCB$L_NT_INTSSB	00000078	D
DDB$L_ACPD	00000010	D		UCB$B_REMSRV	0000003D	D		UCB$L_OPCNT	00000060	D
DDB$L_DDT	0000000C	D		UCB$B_SECTORS	0000003C	D		UCB$L_OWNUIC	0000001C	D
DDB$L_LINK	00000000	D		UCB$B_SLAVE	00000074	D		UCB$L_PID	00000028	D
DDB$L_UCB	00000004	D		UCB$B_SPR	00000075	D		UCB$L_RQBL	00000004	D
DDB$T_DRVNAME	00000024	D		UCB$B_STATE	00000052	D		UCB$L_RQFL	00000000	D
DDB$T_NAME	00000014	D		UCB$B_TRACKS	0000003D	D		UCB$L_SVAPTE	00000068	D
DDB$W_SIZE	00000008	D		UCB$B_TT_CRFILL	0000009D	D		UCB$L_SVPN	00000064	D
DPT$B_ADPTYPE	0000000C	D		UCB$B_TT_DECRF	000000A1	D		UCB$L_TT_DECHAR	000000A8	D
DPT$B_FLAGS	0000000D	D		UCB$B_TT_DELFF	000000A2	D		UCB$L_TT_RDUE	0000008C	D
DPT$B_REFC	0000000B	D		UCB$B_TT_DESPEE	000000A0	D		UCB$L_TT_RTIMOU	000000B8	D
DPT$B_TYPE	0000000A	D		UCB$B_TT_DETYPE	000000A4	D		UCB$L_VCB	00000030	D
DPT$C_LENGTH	0000002A	D		UCB$B_TT_LFFILL	0000009E	D		UCB$T_PARTNER	0000000C	D
DPT$K_LENGTH	0000002A	D		UCB$B_TT_SPEED	0000009C	D		UCB$W_BCNT	0000006E	D
DPT$L_BLINK	00000004	D		UCB$B_TYPE	0000000A	D		UCB$W_BCR	00000096	D
DPT$L_FLINK	00000000	D		UCB$B_VERTSZ	0000003F	D		UCB$W_BOFF	0000006C	D
DPT$T_NAME	0000001E	D		UCB$C_LENGTH	00000074	D		UCB$W_BUFQUO	00000018	D
DPT$W_INITTAB	00000010	D		UCB$C_MB_LENGTH	00000090	D		UCB$W_BYTESTOGO	0000003E	D
DPT$W_MAXUNITS	00000016	D		UCB$C_TT_LENGTH	000000BC	D		UCB$W_CHARGE	0000004A	D
DPT$W_REINITTAB	00000012	D		UCB$K_LENGTH	00000074	D		UCB$W_CYLINDERS	0000003E	D
DPT$W_SIZE	00000008	D		UCB$K_MB_LENGTH	00000090	D		UCB$W_DA	0000008C	D
DPT$W_UCBSIZE	0000000E	D		UCB$K_TT_LENGTH	000000BC	D		UCB$W_DC	0000008E	D
DPT$W_UNLOAD	00000014	D		UCB$L_AMB	00000054	D		UCB$W_DEVBUFSIZ	0000003A	D
DPT$W_VERSION	00000013	D		UCB$L_ASTQBL	00000010	D		UCB$W_DEVSTS	0000005A	D
DYN$C_CRB =	00000005	D		UCB$L_ASTQFL	0000000C	D		UCB$W_DIRSEQ	00000088	D
DYN$C_DDB =	00000006	D		UCB$L_CPID	0000005C	D		UCB$W_DSTADDR	00000018	D
DYN$C_IDB =	00000009	D		UCB$L_CRB	00000020	D		UCB$W_DX_BCR	000000A4	D
DYN$C_UCB =	00000010	D		UCB$L_DDB	00000024	D		UCB$W_ECT	00000090	D
EXECUTE	00000020	R	D	01	UCB$L_DEVCHAR	00000034	D		UCB$W_EC2	00000092	D
IDB$B_TYPE	0000000A		D		UCB$L_DEVDEPEND	0000003C	D		UCB$W_ERRCNT	00000072	D
IDB$C_LENGTH	00000034		D		UCB$L_DPC	00000080	D		UCB$W_FUNC	0000007E	D
IDB$K_LENGTH	00000034		D		UCB$L_DUETIM	0000005C	D		UCB$W_MB_SEED	00000000	D
IDB$L_ADP	00000010		D		UCB$L_DX_BFPNT	0000009C	D		UCB$W_MSGCNT	00000016	D
IDB$L_CSR	00000000		D		UCB$L_DX_BUF	00000098	D		UCB$W_MSGMAX	00000014	D
IDB$L_OWNER	00000004		D		UCB$L_DX_RXDB	000000A0	D		UCB$W_NT_CHAN	0000007C	D
IDB$L_UCBLST	00000014		D		UCB$L_EMB	00000078	D		UCB$W_OFFSET	0000008A	D
IDB$W_SIZE	00000008		D		UCB$L_FIRST	00000014	D		UCB$W_REFC	00000050	D
IDB$W_UNITS	0000000C		D		UCB$L_FPC	0000000C	D		UCB$W_SIZE	00000008	D
IOC$INITDRV	00000004	RG	D	01	UCB$L_FQBL	00000004	D		UCB$W_SRCADDR	0000001A	D
IOC$REINITDRV	0000000A	RG	D	01	UCB$L_FQFL	00000000	D		UCB$W_STS	00000058	D
LOCATE	00000072	R	D	01	UCB$L_FR3	00000010	D		UCB$W_TT_DESIZE	000000A5	D
SIZ... =	00000001		D		UCB$L_FR4	00000014	D		UCB$W_UNIT	00000048	D
STYPE	00000000	R	D	01	UCB$L_IOQBL	00000044	D		UCB$W_VPROT	0000001A	D

VEC$B_DATAPATH	00000013	D
VEC$B_NUMREG	00000012	D
VEC$C_LENGTH	00000024	D
VEC$K_LENGTH	00000024	D
VEC$L_ADP	00000014	D
VEC$L_IDB	00000008	D
VEC$L_INITIAL	0000000C	D
VEC$L_START	0000001C	D
VEC$L_UNITDISC	00000020	D
VEC$L_UNITINIT	00000018	D
VEC$Q_DISPATCH	00000000	D
VEC$W_MAPREG	00000010	D

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
SEP	000000A4 (164.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	000000BC (188.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...			
CRB$L_INTD	00000014			#-244	(1)		
DPT$W_INITTAB	00000010			#-76	(1)		
DPT$W_REINITTAB	00000012			#-104	(1)		
DYN$C_CRB	=00000005			48	(1)		
DYN$C_DDB	=00000006			48	(1)		
DYN$C_IDB	=00000009			48	(1)		
DYN$C_UCB	=00000010			48	(1)		
EXECUTE	00000020-R	135	(1)	#-110	(1)		
IOC$INITDRV	00000004-R	75	(1)				
IOC$REINITDRV	0000000A-R	103	(1)				
LOCATE	00000072-R	218	(1)	#-106	(1)		
STYPE	00000000-R	48	(1)	222	(1)		
UCB$L_CRB	00000020			#-243	(1)	#-258	(1)
UCB$L_DDB	00000024			#-251	(1)		
VEC$L_IDB	00000008			#-244	(1)		

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CRBDEF	1	33 (1)	33 (1)
$DDBDEF	1	34 (1)	34 (1)
$DEFINI	1	33 (1)	33 (1) 34 (1) 35 (1) 36 (1) 37 (1) 38 (1) 39 (1)
$DPTDEF	2	35 (1)	35 (1)
$DYNDEF	2	36 (1)	36 (1)
$IDBDEF	1	37 (1)	37 (1)
$UCBDEF	10	38 (1)	38 (1)
$VECDEF	2	39 (1)	39 (1)
CASE	1	136 (1)	136 (1) 149 (1) 223 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.14	00:00:00.34
Command processing	144	00:00:00.86	00:00:02.00
Pass 1	428	00:00:07.95	00:00:11.77
Symbol table sort	0	00:00:00.42	00:00:00.44
Pass 2	80	00:00:01.47	00:00:01.89
Symbol table output	16	00:00:00.11	00:00:00.11
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	11	00:00:00.11	00:00:00.11
Assembler run totals	720	00:00:11.09	00:00:16.70

The working set limit was 1000 pages.
32358 bytes (64 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 317 non-local and 23 local symbols.
269 source lines were read in Pass 1, producing 0 object records in Pass 2.
70 pages of virtual memory were used to define 17 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	7
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	13

618 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) RELOCDRV/UPDA=(RELOCDRV.UPD,RELOCDRV.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO

```
;  0001  0     %title '*** RMS Master RMS routines'
;  0002  0     module rms (                                    ! Emulate RMS for VAX Supervisor
;  0003  0                     ident = '06.23'
;  0004  0                     ) =
;  0005  0     !
;  0006  0     ! Copyright (c) 1980, 1982, 1983, 1984
;  0007  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;  0008  0     !
;  0009  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;  0010  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;  0011  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;  0012  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;  0013  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;  0014  0     ! TERMS.  TITLE TO AND  OWNERSHIF OF THE  SOFTWARE  SHALL AT ALL TIMES
;  0015  0     ! REMAIN IN DEC.
;  0016  0     !
;  0017  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;  0018  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;  0019  0     ! CORPORATION.
;  0020  0     !
;  0021  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;  0022  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;  0023  0     !
;  0024  0     !++
;  0025  0     ! FACILITY: Diagnostic Supervisor
;  0026  0     !
;  0027  0     ! ABSTRACT:
;  0028  0     !
;  0029  0     !       This module emulates RMS (read-only functions) for the
;  0030  0     !       Supervisor.  It additionally supports RT-11 format files
;  0031  0     !       on console floppy.
;  0032  0     !
;  0033  0     ! AUTHOR:
;  0034  0     !       Dave Butenhof,          4-aug-1980
;  0035  0     !
;  0036  0     ! MODIFIED BY:
;  0037  0     !
;  0038  0     !               Dave Butenhof, 02-oct-1980, Version 6.1
;  0039  0     !       1       Eliminate possiblility of accessing console storage device
;  0040  0     !               as "CS:" by explicitly checking for controller and unit.
;  0041  0     !       2       Make "last-chance" backup filespec global as "DEF$Q_BACKSTOP"
;  0042  0     !       3       <DEBUG> to make user mode debug easier, conditionalize MFPR
;  0043  0     !               to execute only in standalone.
;  0044  0     !       4       Add TS11 rpb setup to RMS$OPEN.
;  0045  0     !               Dave Butenhof, 9-apr-1981, Version 6.3
;  0046  0     !       05      Fix up some context problems in RMS base routines.  Namely,
;  0047  0     !               Return EOF if VBN = eofvbn and ffb = 0 ($READ).  Also,
;  0048  0     !               start RSZ and RBF as 0!!!
;  0049  0     !               - Dave butenhof, 02-Jun-1981, version 6.4
;  0050  0     !       06      Alter directory spec on libraries, use logical names $DS,
;  0051  0     !               $DIAG for flexibility.
;  0052  0     !               - Dave Butenhof, 24-Jun-1981, version 6.4                [07]
;  0053  0     !       07      Setup RPB for console device, to allow using the console [07]
;  0054  0     !               BOOTDRIVR routines.  Also, start cleaning up some of the [07]
;  0055  0     !               Macro JSB routine linkages to use Bliss V2.1's register output [07]
;  0056  0     !               parameter capability, instead of global registers.
;  0057  0     !       08      - Dave Butenhof, 31-aug-1981, version 6.5
```

```
0058  0     !                Support DQ device (IDC) for Nebula, as ODS disk
0059  0     !
0060  0     !        09      - Dave Butenhof, 26-Oct-1981, version 6.-
0061  0     !                Alter call to LOC$PTABLE to use now argument, link address
0062  0     !
0063  0     !        10      - Dave Butenhof, 25-Jan-1982, version 6.6
0064  0     !                Fix calls to Exe$AloNonPaged to use output arguments instead
0065  0     !                of global registers.  Also, add new routines to allocate and
0066  0     !                init pseudo RPB, and to deallocate same.  Generally clean up
0067  0     !                RMS$OPEN somewhat.
0068  0     !
0069  0     !        11      - Dave Butenhof, 05-Mar-1982, version 6.6
0070  0     !                Fix edit 10...can't have size field of data structure be
0071  0     !                direct output parameter of AloNonPaged, since pointer hasn't
0072  0     !                been set yet!
0073  0     !
0074  0     ! [12] Dave Butenhof, 15-Apr-1982, version 6.7
0075  0     !       Add support for UDA-50 disks (DUan)
0076  0     ! [13] Dave Butenhof, 24-May-1982, version 6.8
0077  0     !       Generalize checking for device support with DS$GA_SUPPORT_TABLE
0078  0     !       data structure and DSX$CHECK_SUPPORT routine.  This is only for
0079  0     !       VDS file services, and does not/should not reflect all devices.
0080  0     !
0081  0     !        14      Bob Bergazzi    7-Aug-1982      version 6.9
0082  0     !                Edit 13 had a typo - RL211 in device support table would not
0083  0     !                allow file services for an RL02 connected to an RL11.
0084  0     !
0085  0     !        15      M. Baggett      1-Dec-1982      version 6.10
0086  0     !                Added support for TU80 magtape as load device.
0087  0     !
0088  0     !        16      M. Baggett      7-Dec-1982      version 6.10
0089  0     !                Added support for TU81 magtape as load device.
0090  0     !
0091  0     !        17      M. Baggett      23-Feb-1983     Version 6.11
0092  0     !                Added support for RC25 booting and file operations.
0093  0     !
0094  0     !        18      M. Baggett      1-March-1983    Version 6.11
0095  0     !                Added LESI,RCF25 and support for booting CI-HSC50.
0096  0     !
0097  0     !        19      M. Baggett      17-May-1983     Version 6.11
0098  0     !                Added changes to support CI-HSC50 port station #.
0099  0     !        20      M. Baggett      22-Sep-1983     Version 6.13
0100  0     !                Changed name LESI to RC11
0101  0     !
0102  0     !        21      M. Baggett      11-Oct-1983     Version 6.13
0103  0     !                Added support for CI-HSC50 tape and   05
0104  0     !
0105  0     !        22      Domenic Andella 1-Mar-1984      Version 6.13
0106  0     !                Added code in routines DSX$CHECK_SUPPORT and
0107  0     !                RMS$OPEN for xxx CPU. Each routine uses code
0108  0     !                to 'backtrack' through ptables till _LINK field
0109  0     !                equals zero, when this occurs the present Ptable
0110  0     !                is linked to HUB. Not so for a xxx , when ptable
0111  0     !                _LINK field equals zero it means were linked to
0112  0     !                to an SBIA since the HUB is ABUS.  Therfore the
0113  0     !                modifications have to do with saving and reloading
0114  0     !                the previous adapter and controller address if
```

```
;  0115  0     !                   xxx ,
;  0116  0     !
;  0117  0     !       23      Bob Bergazzi    April 13, 1984  Version 7.0
;  0118  0     !               Changed DSX$Check_support to support ODS format on
;  0119  0     !               the console storage device for the ZZZ cpu.
;  0120  0     !--
;  0121  0
;  0122  1     begin
;  0123  1     !
;  0124  1     ! include files:
;  0125  1     !
;  0126  1
;  0127  1     library '$diag';                !
;  0128  1
;  0129  1     library '$ds';                  !
;  0130  1
;  0131  1     library 'sys$library:lib.l32';
;  0132  1
;  0133  1     !
;  0134  1     ! table of contents:
;  0135  1     !
;  0136  1
;  0137  1     forward routine
;  0138  1         rms$error : jsb_rms,                    ! Load error status
;  0139  1         rms$load_xab : call_rms novalue,        ! Load XAB data
;  0140  1         rms$alloc_cache : jsb_cblock,           ! Allocate cache
;  0141  1         rms$init : jsb_none novalue,            ! Initialize
;  0142  1         rms$cleanup : jsb_none novalue,         ! Clean up RMS allocations
;  0143  1         rab$error : jsb_rms,                    ! Store error code in RAB
;  0144  1         fab$error : jsb_rms,                    ! Store error code in FAB
;  0145  1         verify_cblock : jsb_rms,                ! Be sure IFI and cblock pointer are OK
;  0146  1         verify_rab : jsb_rms,                   ! Verify rab
;  0147  1         verify_fab : jsb_rms,                   ! Verify fab
;  0148  1         rms$open,                               ! RMS$OPEN emulator--open file
;  0149  1         rms$connect,                            ! RMS$CONNECT emulator
;  0150  1         rms$get,                                ! RMS$GET emulator--record de-block
;  0151  1         rms$disconnect,                         ! RMS$DISCONNECT emulator
;  0152  1         rms$close;                              ! RMS$CLOSE emulator--close file
;  0153  1
;  0154  1     !
;  0155  1     ! Local values
;  0156  1     !
;  0157  1
;  0158  1     literal
;  0159  1         True = 1,                                       ! Literal TRUE
;  0160  1         False = 0,                                      ! literal FALSE
;  0161  1         rms$c_maxfiles = 3,                     ! Maximum files open at once
;  0162  1         device$_ansi = dev$m_sqd or dev$m_fod,  ! Describe magtape
;  0163  1         device$_ods = dev$m_fod or dev$m_dir or dev$m_rnd,  ! Describe disk
;  0164  1         device$_rt11 = dev$m_dir or dev$m_rnd or dev$m_sdi or dev$m_fod,    ! Describe floppy
;  0165  1         ! Re-define Restart parameter block length as 16-byte multiple,
;  0166  1         ! to make allocation easier w/out requiring assumptions or
;  0167  1         ! explicitly storing the actual length.
;  0168  1         rpb$_length = (rpb$c_length + 15) and not 15;
;  0169  1
;  0170  1     !
;  0171  1     ! external references:
```

```
;    0172  1      !
;    0173  1      $ds_dsadef;
;    0174  1
;    0175  1      external
;    0176  1          Ds$Gb_Uda50_Init : Byte,                          ! Global init flag in PUBTDRIVR
;    0177  1          Ds$Gb_CI_Init : Byte Weak,                        ! Global init flag in PABTDRIVR
;    0178  1          def$q_dir : bblock addressing_mode (long_relative),
;    0179  1          def$q_dev : bblock addressing_mode (long_relative),
;    0180  1          boo$select,                                       ! Non-interrupt "QIO" package stuff
;    0181  1          boo$al_vector : addressing_mode (long_relative);
;    0182  1
;    0183  1      external routine
;    0184  1          exe$deanonpaged : jsb_deall addressing_mode (long_relative),      ! Deallocate dyn. pool space
;    0185  1          exe$alononpaged : jsb_alloc addressing_mode (long_relative),      ! Allocate from dynamic pool
;    0186  1          fil$cvtfilnam,                                    ! Convert ascii to RAD50
;    0187  1          BreakUP_File,
;    0188  1          loc$ptable : jsb_loc$ptable novalue addressing_mode (long_relative),
;    0189  1          rt11$open : call_rms,                             ! Access console RT-11 file
;    0190  1          ods$open : call_rms,                              ! Access disk file
;    0191  1          ansi$open : call_rms addressing_mode (long_relative);      ! Access magtape file
;    0192  1
;    0193  1      !
;    0194  1      ! psect definitions:
;    0195  1      !
;    0196  1
;    0197  1      psect
;    0198  1          plit = data ( share, addressing_mode (long_relative));
;    0199  1
;    0200  1      psect
;    0201  1          own = work ( read, write, addressing_mode (long_relative));
;    0202  1
;    0203  1      psect
;    0204  1          global = work ( read, write, addressing_mode (long_relative));
;    0205  1
;    0206  1      psect
;    0207  1          code = code ( read, execute, share);
;    0208  1
;    0209  1      !
;    0210  1      ! Own storage
;    0211  1      !
;    0212  1
;    0213  1      global
;    0214  1          rms$file_table : vector [4];                  ! ifi-indexed pointers
;    0215  1
;    0216  1      !
;    0217  1      ! Global data definitions:
;    0218  1      !
;    0219  1
;    0220  1      global bind
;    0221  1          def$c_bkstp_len = %charcount (%ascii'CSA0:[SYSMAINT];0'),
;    0222  1          def$q_backstop = uplit byte(%ascii'CSA0:[SYSMAINT];0');
;    0223  1
```

```
1      %Sbttl 'Device support table'
1
1      Compiletime                                                  !                                                        [13]
1          $$_Rows_$$ = 0;                                          !                                                        [13]
1
1      Global Bind                                                  ! Device support table                                   [13]
1          Ds$GA_Support_Table = Uplit (                            !                                                        [13]
1              Support_Entry ('console', HUB, Btd$K_Console, RT11, <>, 0, <>),                              ! Console        [13]
1              Support_Entry ('console', HUB, Btd$K_Console, ODS, <>, 0, <>),                               ! ZZZ console    [23]
1              Support_Entry ('RP04', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RP04           [13]
1              Support_Entry ('RP05', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RP05           [13]
1              Support_Entry ('RP06', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RP06           [13]
1              Support_Entry ('RP07', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RP07           [13]
1              Support_Entry ('RM03', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RM03           [13]
1              Support_Entry ('RM05', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RM05           [13]
1              Support_Entry ('RM80', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                     ! RM80           [13]
1              Support_Entry ('TM03', MBA, Btd$K_Tm, ANSI, <A, C>, 0, <'TE16', 'TU45', 'TU77'>),          ! TM03           [13]
1              Support_Entry ('TM78', MBA, Btd$K_Tf, ANSI, <A, C>, 0, <'TU78'>),                           ! TM78           [13]
1              Support_Entry ('RB730',UBA, Btd$K_Dq, ODS, <A, C>, 0, <'RL02', 'R80'>),                     ! IDC            [13]
1              Support_Entry ('UDA50',UBA, Btd$K_UDA,ODS, <A, C>, Ds$GB_Uda50_Init, <'RA80','RA81','RA60'>), !UDA50          [18]
1              Support_Entry ('RK611',UBA, Btd$K_Dm, ODS, <A, C>, 0, <'RK06', 'RK07'>),                    ! RK06/07        [13]
1              Support_Entry ('RL11' ,UBA, Btd$K_Dl, ODS, <A, C>, 0, <'RL02'>),                            ! RL02           [14]
1              Support_Entry ('TAPE', UBA, Btd$K_UDA, ANSI,<A>, 0, <>),                                    ! HSC50 TAPE     [21]
1              Support_Entry ('TS11', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                     ! TS04           [13]
1              Support_Entry ('TS04', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                     ! TS04           [13]
1              Support_Entry ('TS05', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                     ! TS05           [21]
1              Support_Entry ('TU80', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                     ! TU80           [15]
1              Support_Entry ('TU81', UBA, Btd$K_Mu, ANSI,<A>, 0, <>),                                     ! TU81           [17]
1              Support_Entry ('CI_NODE',CIA, Btd$K_HSCCI,ODS, <A, C>, DS$GB_CI_INIT, <'DISK'>),             ! HSC50          [20]
1              Support_Entry ('RCT1',UBA, Btd$K_UDA,ODS, <A, C>, Ds$GB_Uda50_Init, <'RC25','RCF25'>)       ! UDA50          [12]
1              );                                           !                                                        [13]
1
1      Global Literal                                               !                                                        [13]
1          Ds$GK_Support_Table_Rows = $$_Rows_$$;                   !                                                        [13]
```

```
;    0258  1     %Sbttl 'Allocate pseudo RPB'
;    0259  1
;    0260  1     !++
;    0261  1     ! Functional description:
;    0262  1     !       Allocate a block of nonpaged pool and initialize it to resemble an
;    0263  1     !       RPB data structure.  Note that it is NOT an RPB for a number of
;    0264  1     !       reasons, including the fact that it may not be page-aligned.  It is
;    0265  1     !       close enough to an RPB to make the VMS standalone BOOTDRIVR I/O
;    0266  1     !       mechanism function correctly.
;    0267  1     !
;    0268  1     ! Input Parameters:
;    0269  1     !       04(Ap)          DeviceType (type code for device [DM, DL, MB,...])
;    0270  1     !       08(Ap)          ADPType (type code for adapter [MBA, UBA])
;    0271  1     !       12(Ap)          ADPPhy (physical/virtual address of adapter)
;    0272  1     !       16(Ap)          CSRPhy (physical/virtual address of controller)
;    0273  1     !       20(Ap)          Drive (Unit number of drive)
;    0274  1     !       24(AP)          Slave unit number
;    0275  1     !       28(AP)          Port number on CI network for HSC50.                    [19]
;    0276  1     !
;    0277  1     ! Outputs:
;    0278  1     !       R0              0 if couldn't allocate RPB; else address of RPB
;    0279  1     !--
;    0280  1
;    0281  1     Global Routine Dsr$Allocate_Pseudo_RPB (        !                               [10]
;    0282  1             DeviceType, ADPType, CSRPhy, ADPPhy, Drive, Slave, Ciport) =    !       [10]
;    0283  2         Begin                       !                                           [10]
;    0284  2
;    0285  2         Builtin                     !                                           [10]
;    0286  2             Mfpr;                   !                                           [10]
;    0287  2
;    0288  2         Local                       !                                           [10]
;    0289  2             RPB : Ref Block [, Byte];       !                                   [10]
;    0290  2
;    0291  2         If Not Exe$AloNonPaged (RPB$_Length;, RPB)  !                           [10]
;    0292  2         Then                        !                                           [10]
;    0293  2             Return 0;               !                                           [10]
;    0294  2
;    0295  2         Ch$fill (0, RPB$_Length, .RPB);     ! Zero it for good measure          [10]
;    0296  2         .RPB = .RPB;                ! First longword is own address         [10]
;    0297  2         RPB [RPB$B_DevTyp] = .DeviceType;   !                                   [10]
;    0298  2         RPB [RPB$L_BootR1] = 0;     !                                           [10]
;    0299  2         RPB [RPB$L_IOVec] = Boo$AL_Vector;  !                                   [10]
;    0300  2         (RPB [RPB$B_ConfReg])<0, 8> = .ADPType;     !                           [10]
;    0301  2         RPB [RPB$L_CSRPhy] = RPB [RPB$L_CSRVir] = .CSRPhy;  !                   [10]
;    0302  2         RPB [RPB$L_ADPPhy] = RPB [RPB$L_ADPVir] = .ADPPhy;  !                   [10]
;    0303  2         RPB [RPB$W_Unit] = .Drive;  !                                           [10]
;    0304  2         RPB [RPB$B_Slave] = .Slave; !                                           [10]
;    0305  2         RPB [RPB$L_BOOTR2] = .Ciport; ! HSC50 CI port number.                   [19]
;    0306  2         Boo$AL_Vector + 8 = Boo$Select - Boo$AL_Vector;     ! Initialize the entry.  [10]
;    0307  2
;    0308  2         If Not .Dsa$V_User Then Mfpr (Pr$_Sbr, RPB [RPB$L_Svaspt]); ! Get sys page table addr.  [10]
;    0309  2
;    0310  2         .RPB                        !                                           [10]
;    0311  1         End;                        !                                           [10]

                                                .TITLE  RMS *** RMS Master RMS routines
```

```
                                                        .IDENT  \06.23\

                                                        .PSECT  WORK,NOEXE,2

                                                00000 RMS$FILE_TABLE::
                                                        .BLKB   16

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

5D  54  4E  49  41  4D  53  59  53  5B  3A  30  41  53  43  00000 P.AAA:   .ASCII  \CSA0:[SYSMAINT];0\                    ;
                                                    30  3B  0000F
                            65  6C  6F  73  6E  6F  63  07  00011 P.AAC:   .ASCII  <7>\console\                         ;
                            65  6C  6F  73  6E  6F  63  07  00019 P.AAD:   .ASCII  <7>\console\                         ;
                                34  30  50  52  04  00021 P.AAE:   .ASCII  <4>\RP04\                            ;
                                35  30  50  52  04  00026 P.AAF:   .ASCII  <4>\RP05\                            ;
                                36  30  50  52  04  0002B P.AAG:   .ASCII  <4>\RP06\                            ;
                                37  30  50  52  04  00030 P.AAH:   .ASCII  <4>\RP07\                            ;
                                33  30  4D  52  04  00035 P.AAI:   .ASCII  <4>\RM03\                            ;
                                35  30  4D  52  04  0003A P.AAJ:   .ASCII  <4>\RM05\                            ;
                                30  38  4D  52  04  0003F P.AAK:   .ASCII  <4>\RM80\                            ;
                                33  30  4D  54  04  00044 P.AAL:   .ASCII  <4>\TM03\                            ;
                                36  31  45  54  04  00049 P.AAN:   .ASCII  <4>\TE16\                            ;
                                35  34  55  54  04  0004E P.AAO:   .ASCII  <4>\TU45\                            ;
                                37  37  55  54  04  00053 P.AAP:   .ASCII  <4>\TU77\                            ;
                    00000000'   00000000'   00000000'   00058 P.AAM:   .ADDRESS P.AAN, P.AAO, P.AAP                ;
                                    00000000            00064         .LONG   0                                   ;
                                38  37  4D  54  04  00068 P.AAQ:   .ASCII  <4>\TM78\                            ;
                                38  37  55  54  04  0006D P.AAS:   .ASCII  <4>\TU78\                            ;
                                                    00072         .BLKB   2
                                    00000000'       00074 P.AAR:   .ADDRESS P.AAS                               ;
                                    00000000        00078         .LONG   0                                   ;
                        30  33  37  42  52  05  0007C P.AAT:   .ASCII  <5>\RB730\                           ;
                            32  30  4C  52  04  00082 P.AAV:   .ASCII  <4>\RL02\                            ;
                                30  38  52  03  00087 P.AAW:   .ASCII  <3>\R80\                             ;
                                                    0008B         .BLKB   1
                        00000000'   00000000'       0008C P.AAU:   .ADDRESS P.AAV, P.AAW                        ;
                                    00000000        00094         .LONG   0                                   ;
                        30  35  41  44  55  05  00098 P.AAX:   .ASCII  <5>\UDA50\                           ;
                            30  38  41  52  04  0009E P.AAZ:   .ASCII  <4>\RA80\                            ;
                            31  38  41  52  04  000A3 P.ABA:   .ASCII  <4>\RA81\                            ;
                            30  36  41  52  04  000A8 P.ABB:   .ASCII  <4>\RA60\                            ;
                                                    000AD         .BLKB   3
    00000000'   00000000'   00000000'   000B0 P.AAY:   .ADDRESS P.AAZ, P.ABA, P.ABB                        ;
                                    00000000        000BC         .LONG   0                                   ;
                        31  31  36  4B  52  05  000C0 P.ABC:   .ASCII  <5>\RK611\                           ;
                            36  30  4B  52  04  000C6 P.ABE:   .ASCII  <4>\RK06\                            ;
                            37  30  4B  52  04  000CB P.ABF:   .ASCII  <4>\RK07\                            ;
                            00000000'   00000000'   000D0 P.ABD:   .ADDRESS P.ABE, P.ABF                        ;
                                    00000000        000D8         .LONG   0                                   ;
                            31  31  4C  52  04  000DC P.ABG:   .ASCII  <4>\RL11\                            ;
                            32  30  4C  52  04  000E1 P.ABI:   .ASCII  <4>\RL02\                            ;
                                                    000E6         .BLKB   2
                                    00000000'       000E8 P.ABH:   .ADDRESS P.ABI                               ;
                                    00000000        000EC         .LONG   0                                   ;
                            45  50  41  54  04  000F0 P.ABJ:   .ASCII  <4>\TAPE\                            ;
                            31  31  53  54  04  000F5 P.ABK:   .ASCII  <4>\TS11\                            ;
                            34  30  53  54  04  000FA P.ABL:   .ASCII  <4>\TS04\                            ;
```

```
                                35  30  53  54  04  000FF P.ABM:   .ASCII  <4>\TS05\                                ;
                                30  38  55  54  04  00104 P.ABN:   .ASCII  <4>\TU80\                                ;
                                31  38  55  54  04  00109 P.ABO:   .ASCII  <4>\TU81\                                ;
                            45  44  4F  4E  5F  49  43  07  0010E P.ABP:   .ASCII  <7>\CI_NODE\                       ;
                                4B  53  49  44  04  00116 P.ABR:   .ASCII  <4>\DISK\                                ;
                                                    0011B         .BLKB   1                                         ;
                                            00000000' 0011C P.ABQ:   .ADDRESS P.ABR                                 ;
                                            00000000  00120         .LONG   0                                       ;
                                31  31  43  52  04  00124 P.ABS:   .ASCII  <4>\RC11\                                ;
                                35  32  43  52  04  00129 P.ABU:   .ASCII  <4>\RC25\                                ;
                            35  32  46  43  52  05  0012E P.ABV:   .ASCII  <5>\RCF25\                                ;
                            00000000' 00000000' 00134 P.ABT:   .ADDRESS P.ABU, P.ABV                                ;
                                            00000000  0013C         .LONG   0                                       ;
                                            00000000' 00140 P.AAB:   .ADDRESS P.AAC                                 ;
                                                  00  00144         .BYTE   0                                       ;
                                                  40  00145         .BYTE   64                                      ;
                                                  03  00146         .BYTE   3                                       ;
                                                  00  00147         .BYTE   0                                       ;
                                            00000000  00148         .LONG   0                                       ;
                                            00000000  0014C         .LONG   0                                       ;
                                            00000000' 00150         .ADDRESS P.AAD                                  ;
                                                  00  00154         .BYTE   0                                       ;
                                                  40  00155         .BYTE   64                                      ;
                                                  02  00156         .BYTE   2                                       ;
                                                  00  00157         .BYTE   0                                       ;
                                            00000000  00158         .LONG   0                                       ;
                                            00000000  0015C         .LONG   0                                       ;
                                            00000000' 00160         .ADDRESS P.AAE                                  ;
                                                  20  00164         .BYTE   32                                      ;
                                                  00  00165         .BYTE   0                                       ;
                                                  02  00166         .BYTE   2                                       ;
                                                  01  00167         .BYTE   1                                       ;
                                            00000000  00168         .LONG   0                                       ;
                                            00000000  0016C         .LONG   0                                       ;
                                            00000000' 00170         .ADDRESS P.AAF                                  ;
                                                  20  00174         .BYTE   32                                      ;
                                                  00  00175         .BYTE   0                                       ;
                                                  02  00176         .BYTE   2                                       ;
                                                  01  00177         .BYTE   1                                       ;
                                            00000000  00178         .LONG   0                                       ;
                                            00000000  0017C         .LONG   0                                       ;
                                            00000000' 00180         .ADDRESS P.AAG                                  ;
                                                  20  00184         .BYTE   32                                      ;
                                                  00  00185         .BYTE   0                                       ;
                                                  02  00186         .BYTE   2                                       ;
                                                  01  00187         .BYTE   1                                       ;
                                            00000000  00188         .LONG   0                                       ;
                                            00000000  0018C         .LONG   0                                       ;
                                            00000000' 00190         .ADDRESS P.AAH                                  ;
                                                  20  00194         .BYTE   32                                      ;
                                                  00  00195         .BYTE   0                                       ;
                                                  02  00196         .BYTE   2                                       ;
                                                  01  00197         .BYTE   1                                       ;
                                            00000000  00198         .LONG   0                                       ;
                                            00000000  0019C         .LONG   0                                       ;
                                            00000000' 001A0         .ADDRESS P.AAI                                  ;
                                                  20  001A4         .BYTE   32                                      ;
```

```
                00  001A5          .BYTE   0                                       ;
                02  001A6          .BYTE   2                                       ;
                01  001A7          .BYTE   1                                       ;
        00000000    001A8          .LONG   0                                       ;
        00000000    001AC          .LONG   0                                       ;
        00000000'   001B0          .ADDRESS P.AAJ                                  ;
                20  001B4          .BYTE   32                                      ;
                00  001B5          .BYTE   0                                       ;
                02  001B6          .BYTE   2                                       ;
                01  001B7          .BYTE   1                                       ;
        00000000    001B8          .LONG   0                                       ;
        00000000    001BC          .LONG   0                                       ;
        00000000'   001C0          .ADDRESS P.AAK                                  ;
                20  001C4          .BYTE   32                                      ;
                00  001C5          .BYTE   0                                       ;
                02  001C6          .BYTE   2                                       ;
                01  001C7          .BYTE   1                                       ;
        00000000    001C8          .LONG   0                                       ;
        00000000    001CC          .LONG   0                                       ;
        00000000'   001D0          .ADDRESS P.AAL                                  ;
                20  001D4          .BYTE   32                                      ;
                B1  001D5          .BYTE   -79                                     ;
                01  001D6          .BYTE   1                                       ;
                03  001D7          .BYTE   3                                       ;
        00000000    001D8          .LONG   0                                       ;
        00000000'   001DC          .ADDRESS P.AAM                                  ;
        00000000'   001E0          .ADDRESS P.AAQ                                  ;
                20  001E4          .BYTE   32                                      ;
                B3  001E5          .BYTE   -77                                     ;
                01  001E6          .BYTE   1                                       ;
                03  001E7          .BYTE   3                                       ;
        00000000    001E8          .LONG   0                                       ;
        00000000'   001EC          .ADDRESS P.AAR                                  ;
        00000000'   001F0          .ADDRESS P.AAT                                  ;
                00  001F4          .BYTE   0                                       ;
                03  001F5          .BYTE   3                                       ;
                02  001F6          .BYTE   2                                       ;
                03  001F7          .BYTE   3                                       ;
        00000000    001F8          .LONG   0                                       ;
        00000000'   001FC          .ADDRESS P.AAU                                  ;
        00000000'   00200          .ADDRESS P.AAX                                  ;
                00  00204          .BYTE   0                                       ;
                11  00205          .BYTE   17                                      ;
                02  00206          .BYTE   2                                       ;
                03  00207          .BYTE   3                                       ;
        00000000G   00208          .ADDRESS DS$GB_UDA50_INIT                       ;
        00000000'   0020C          .ADDRESS P.AAY                                  ;
        00000000'   00210          .ADDRESS P.ABC                                  ;
                00  00214          .BYTE   0                                       ;
                01  00215          .BYTE   1                                       ;
                02  00216          .BYTE   2                                       ;
                03  00217          .BYTE   3                                       ;
        00000000    00218          .LONG   0                                       ;
        00000000'   0021C          .ADDRESS P.ABD                                  ;
        00000000'   00220          .ADDRESS P.ABG                                  ;
                00  00224          .BYTE   0                                       ;
                02  00225          .BYTE   2                                       ;
```

```
                02  00226          .BYTE   2                                                                       ;
                03  00227          .BYTE   3                                                                       ;
          00000000  00228          .LONG   0                                                                       ;
          00000000' 00022C         .ADDRESS P.ABH                                                                  ;
          00000000' 00230          .ADDRESS P.ABJ                                                                  ;
                00  00234          .BYTE   0                                                                       ;
                11  00235          .BYTE   17                                                                      ;
                01  00236          .BYTE   1                                                                       ;
                01  00237          .BYTE   1                                                                       ;
          00000000  00238          .LONG   0                                                                       ;
          00000000  0023C          .LONG   0                                                                       ;
          00000000' 00240          .ADDRESS P.ABK                                                                  ;
                00  00244          .BYTE   0                                                                       ;
                B2  00245          .BYTE   -78                                                                     ;
                01  00246          .BYTE   1                                                                       ;
                01  00247          .BYTE   1                                                                       ;
          00000000  00248          .LONG   0                                                                       ;
          00000000  0024C          .LONG   0                                                                       ;
          00000000' 00250          .ADDRESS P.ABL                                                                  ;
                00  00254          .BYTE   0                                                                       ;
                B2  00255          .BYTE   -78                                                                     ;
                01  00256          .BYTE   1                                                                       ;
                01  00257          .BYTE   1                                                                       ;
          00000000  00258          .LONG   0                                                                       ;
          00000000  0025C          .LONG   0                                                                       ;
          00000000' 00260          .ADDRESS P.ABM                                                                  ;
                00  00264          .BYTE   0                                                                       ;
                B2  00265          .BYTE   -78                                                                     ;
                01  00266          .BYTE   1                                                                       ;
                01  00267          .BYTE   1                                                                       ;
          00000000  00268          .LONG   0                                                                       ;
          00000000  0026C          .LONG   0                                                                       ;
          00000000' 00270          .ADDRESS P.ABN                                                                  ;
                00  00274          .BYTE   0                                                                       ;
                B2  00275          .BYTE   -78                                                                     ;
                01  00276          .BYTE   1                                                                       ;
                01  00277          .BYTE   1                                                                       ;
          00000000  00278          .LONG   0                                                                       ;
          00000000  0027C          .LONG   0                                                                       ;
          00000000' 00280          .ADDRESS P.ABO                                                                  ;
                00  00284          .BYTE   0                                                                       ;
                B4  00285          .BYTE   -76                                                                     ;
                01  00286          .BYTE   1                                                                       ;
                01  00287          .BYTE   1                                                                       ;
          00000000  00288          .LONG   0                                                                       ;
          00000000  0028C          .LONG   0                                                                       ;
          00000000' 00290          .ADDRESS P.ABP                                                                  ;
                00  00294          .BYTE   0                                                                       ;
                20  00295          .BYTE   32                                                                      ;
                02  00296          .BYTE   2                                                                       ;
                03  00297          .BYTE   3                                                                       ;
          00000000G 00298          .ADDRESS DS$GB_CI_INIT                                                          ;
          00000000' 0029C          .ADDRESS P.ABQ                                                                  ;
          00000000' 002A0          .ADDRESS P.ABS                                                                  ;
                00  002A4          .BYTE   0                                                                       ;
                11  002A5          .BYTE   17                                                                      ;
                02  002A6          .BYTE   2                                                                       ;
```

```
                        03  002A7          .BYTE   3                                                       ;
00000000G               002A8          .ADDRESS DS$GB_UDA50_INIT                                       ;
00000000'               002AC          .ADDRESS P.ABT                                                  ;

                                DSA$AL_APTMAIL=     65024
                                DSA$GL_FLAGS=       65024
                                DSA$GL_APTCOM=      65028
                                DSA$GL_PASSES=      65032
                                DSA$GL_UNITS=       65036
                                DSA$GL_SECTNO=      65040
                                DSA$GL_SID=         65044
                                DSA$GL_MSGTYP=      65088
                                DSA$GL_ERRNO=       65092
                                DSA$GL_EVENT=       65096
                                DSA$GL_SUBTNO=      65100
                                DSA$GL_TESTNO=      65104
                                DSA$GL_PASSNO=      65108
                                DSA$GL_DEVLEN=      65112
                                DSA$GL_DEVNAM=      65116
                                DSA$GL_MSGPTR=      65128
                                DEF$C_BKSTP_LEN==   17
                                DEF$Q_BACKSTOP==    P.AAA
                                DS$GA_SUPPORT_TABLE==
                                                    P.AAB
                                DS$GK_SUPPORT_TABLE_ROWS==
                                                    23
                                        .EXTRN  DS$GB_UDA50_INIT
                                        .EXTRN  DEF$Q_DIR, DEF$Q_DEV
                                        .EXTRN  BOO$SELECT, BOO$AL_VECTOR
                                        .EXTRN  EXE$DEANONPAGED
                                        .EXTRN  EXE$ALONONPAGED
                                        .EXTRN  FIL$CVTFILNAM, BREAKUP_FILE
                                        .EXTRN  LOC$PTABLE, RT11$OPEN
                                        .EXTRN  ODS$OPEN, ANSI$OPEN
                                        .WEAK   DS$GB_CI_INIT

                                        .PSECT  CODE,NOWRT,  SHR,2

                                0FFC 00000      .ENTRY  DSR$ALLOCATE_PSEUDO_RPB, Save R2,R3,R4,R5,- ; 0281
                                                        R6,R7,R8,R9,R10,R11                         ;
                        51      0110    8F  3C 00002    MOVZWL  #272, R1                            ; 0291
                            00000000G   EF  16 00007    JSB     EXE$ALONONPAGED                     ;
                        56              52  D0 0000D    MOVL    R2, R6                              ;
                        03              50  E8 00010    BLBS    R0, 1$                              ;
                                        50  D4 00013    CLRL    R0                                  ; 0293
                                        04  00015    RET                                         ;
0110    8F      00      6E              00  2C 00016 1$: MOVC5   #0, (SP), #0, #272, (RPB)           ; 0295
                                        66      0001D                                               ;
                        66              56  D0 0001E    MOVL    RPB, (RPB)                          ; 0296
                    66  A6      04      AC  90 00021    MOVB    DEVICETYPE, 102(RPB)                ; 0297
                        20      A6      D4 00026    CLRL    32(RPB)                             ; 0298
                    34  A6  00000000G   EF  9E 00029    MOVAB   BOO$AL_VECTOR, 52(RPB)              ; 0299
                  0090  C6      08      AC  90 00031    MOVB    ADPTYPE, 144(RPB)                   ; 0300
                        50      0C      AC  D0 00037    MOVL    CSRPHY, R0                          ; 0301
                    58  A6              50  D0 0003B    MOVL    R0, 88(RPB)                         ;
                    54  A5              50  D0 0003F    MOVL    R0, 84(RPB)                         ;
                        50      10      AC  D0 00043    MOVL    ADPPHY, R0                          ; 0302
```

```
                            60   A6              50   D0 00047          MOVL    R0, 96(RPB)                                    ;
                            5C   A6              50   D0 0004B          MOVL    R0, 92(RPB)                                    ;
                            64   A6        14    AC   B0 0004F          MOVW    DRIVE, 100(RPB)                                ; 0303
                            67   A6        18    AC   90 00054          MOVB    SLAVE, 103(RPB)                                ; 0304
                            24   A6        1C    AC   D0 00059          MOVL    CIPORT, 36(RPB)                                ; 0305
                  00000000G  EF 00000000*   8F   D0 0005E              MOVL    #<BOO$SELECT-BOO$AL_VECTOR>, -                 ; 0306
                                                                                 BOO$AL_VECTOR+8                                ;
               04 0000FE03   9F              04   E0 00069              BBS     #4, @#^X0000FE03, 2$                           ; 0308
                      50   A6              0C   DB 00071              MFPR    #12, 80(RPB)                                   ;
                           50              56   D0 00075 2$:          MOVL    RPB, R0                                        ; 0311
                                           04 00078                  RET                                                    ;

; Routine Size:  121 bytes,    Routine Base:  CODE + 0000
```

```
;   0312  1     %Sbttl 'Deallocate RPB'
;   0313  1
;   0314  1     Global Routine Dsr$Deallocate_Pseudo_RPB (RPB) =        !                                  [10]
;   0315  2         Begin                                               !                                  [10]
;   0316  2         (.RPB + 8)<0, 16> = RPB$_Length;                    !                                  [10]
;   0317  2         Exe$DeaNonPaged (.RPB)                              !                                  [10]
;   0318  1         End;                                                !                                  [10]



                                        0FFC 00000          .ENTRY  DSR$DEALLOCATE_PSEUDO_RPB, Save R2,R3,R4,-  ; 0314
                                                                    R5,R6,R7,R8,R9,R10,R11                  ;
                              50    04   AC   D0 00002      MOVL    RPB, R0                                    ; 0316
                        08    A0  0110   8F   B0 00006      MOVW    #272, 8(R0)                                ;
                            00000000G  EF   16 0000C      JSB     EXE$DEANONPAGED                            ; 0317
                                       04 00012           RET                                                ; 0318

; Routine Size:  19 bytes,    Routine Base:  CODE + 0079
```

```
;    0319  1     %Sbttl 'Check device support tables'
;    0320  1
;    0321  1     !+
;    0322  1     ! Function:
;    0323  1     !       Search device support tables for the controller/device specified: return various
;    0324  1     !       information on the pair.
;    0325  1     !
;    0326  1     ! Calling sequence:
;    0327  1     !       status = DSX$CHECK_SUPPORT (Device, Adapter, Btd, File, Init)
;    0328  1     !
;    0329  1     ! Inputs:
;    0330  1     !       Device          => Address of device Ptable
;    0331  1     !
;    0332  1     ! Outputs:
;    0333  1     !       Adc             => Address to receive adapter code (BOOTDRIVR)
;    0334  1     !       Btd             => Address to receive boot device type code
;    0335  1     !       File            => Address to receive file structure type code
;    0336  1     !       Init            => Address to recieve address of driver init flag byte
;    0337  1     !
;    0338  1     ! Status:
;    0339  1     !       SS$_Normal      => OK
;    0340  1     !       DS$_Error       => Device is not supported by file services
;    0341  1     !       DS$_ProgErr     => Error in support table (SPT$V_CONTROLLER set but no drive name list, or bad file structur
;    0342  1     !-
;    0343  1
;    0344  1     Global Routine DSX$Check_Support (Device, Adc, Btd, File, Init) =       !                                          [13]
;    0345  2         Begin                                                               !                                          [13]
;    0346  2                                                                             !                                          [13]
;    0347  2         Builtin                                                             !                                          [13]
;    0348  2             NullParameter;                                                  ! Determine if parameter was specified     [13]
;    0349  2                                                                             !                                          [13]
;    0350  2         Map                                                                 !                                          [13]
;    0351  2             Device : Ref $Ds_Hpo_Decl ();                                   !                                          [13]
;    0352  2                                                                             !                                          [13]
;    0353  2         Local                                                               !                                          [13]
;    0354  2             Controller : Ref $Ds_Hpo_Decl (),                               ! Pointer to controller Ptable             [13]
;    0355  2             Adapter : Ref $DS_Hpo_Decl (),                                  ! Pointer to adapter Ptable                [13]
;    0356  2             Check_Device : Ref Vector [, Byte],                             ! Pointer for controller device name       [13]
;    0357  2             Support_Row : Ref Support_Entry_Def,                            ! Pointer to support list vector           [13]
;    0358  2             Found_Device : Byte Initial (0),                                ! flag for search
;    0359  2             PREV_ADAPTER : REF $DS_HPO_DECL (),                             ! Storage for saved copy of adapter ptable [22]
;    0360  2             PREV_CONTROLLER : REF $DS_HPO_DECL ();                          ! Storage for saved copy of controller ptable [22]
;    0361  2             Adapter = Controller = PREV_ADAPTER = PREV_CONTROLLER = .Device;! Copy Ptable address                       [22]
;    0362  2                                                                             !                                          [13]
;    0363  2         While .Adapter [Hp$A_Link] NeqA 0 Do                                ! Locate link Ptables                      [13]
;    0364  3             Begin                                                           !                                          [13]
;    0365  3             PREV_CONTROLLER = .CONTROLLER;                                  ! Save copy                                [22]
;    0366  3             Controller = .Adapter;                                          ! Shift links down                         [13]
;    0367  3             PREV_ADAPTER =.ADAPTER;                                         ! Save copy                                [22]
;    0368  3             Adapter = .Adapter [Hp$A_Link]                                  !                                          [13]
;    0369  2             End;                                                            !                                          [13]
;    0370  3       BEGIN                                                                 !                                          [22]
;    0371  3         BIND
;    0372  3             DEV_TYPE = ADAPTER [HP$T_TYPE] : VECTOR [, BYTE];       ! Points to ptable _LINK field                     [22]
;    0373  3
;    0374  3     ! The following IF statement checks if the ptable _TYPE field is an SBIA, if so the previously saved adapter       [22]
;    0375  3     ! address is loaded into ADAPTER. This is necessary for a xxx CPU since HUB is equal to ABUS , and a _LINK       [22]
```

```
;    0376  3     ! field of zero would indicate that the present ptable is for an SBIA.                                [22]
;    0377  3
;    0378  3           IF .DEV_TYPE [0] EQL 4                          ! Check for correct ASCII count of 4.          [22]
;    0379  3           AND .(DEV_TYPE [1]) <0,32> EQL %ASCII 'SBIA'    ! Check for _TYPE filed of SBIA                [22]
;    0380  3           THEN                                            !                                              [22]
;    0381  4             BEGIN                                         !                                              [22]
;    0382  4                   CONTROLLER = .PREV_CONTROLLER;          ! Load 'previous' controller ptable address    [22]
;    0383  4                   ADAPTER = .PREV_ADAPTER;                ! Load 'prevous' adapter ptable address        [22]
;    0384  4             END                                           !                                              [22]
;    0385  2       END;                                                !                                              [22]
;    0386  2         Check_Device = Controller [Hp$T_Type];            ! Point to device type of controller           [13]
;    0387  2         Support_Row = Ds$GA_Support_Table;                ! Point to beginning of table                  [13]
;    0388  2
;    0389  2         Incr I From 0 to Ds$GK_Support_Table_Rows - 1 Do  ! Loop through the table                       [13]
;    0390  3             Begin                                         !                                              [13]
;    0391  3
;    0392  3             LITERAL                                       !                                              [23]
;    0393  3                   PR$_SID_TYPZZZ = 5;                     !                                              [23]
;    0394  3
;    0395  3             MAP                                           !                                              [23]
;    0396  3                   DSA$GL_SID : BLOCK;                     !                                              [23]
;    0397  3
;    0398  3             Bind                                          !                                              [13]
;    0399  3                 Device_Type = .Support_Row [Spt$L_Type] : Vector [, Byte];        !                      [13]
;    0400  3
;    0401  3             If .Device_Type [0] Eql .Check_Device [0]      ! If device type lengths are same              [13]
;    0402  3              And Ch$Eql (                                 ! .. and names match                           [13]
;    0403  3                 .Device_Type [0],                         !                                              [13]
;    0404  3                 Device_Type [1],                          !                                              [13]
;    0405  3                 .Device_Type [0],                         !                                              [13]
;    0406  3                 Check_Device [1]                          !                                              [13]
;    0407  3                 )                                         !                                              [13]
;    0408  3             Then                                          !                                              [13]
;    0409  4                 Begin                                     ! Tentative match...verify it                  [13]
;    0410  4
;    0411  4                 Local                                     !                                              [13]
;    0412  4                     Found_Adapter : Byte Initial (0),     ! Flag for matching adapter                    [13]
;    0413  4                     Found_Unit : Byte Initial (0);        ! Flag for matching unit                       [13]
;    0414  4
;    0415  4                 If .Support_Row [Spt$V_Adapter]           ! If there should be an adapter                [13]
;    0416  4                 Then                                      !                                              [13]
;    0417  5                     Begin                                 !                                              [13]
;    0418  5
;    0419  5                     Bind                                  ! Bind to the adapter device type              [13]
;    0420  5                         Adapter_Type = Adapter [Hp$T_Type] : Vector [, Byte];      !                      [13]
;    0421  5
;    0422  5                     If .Support_Row [Spt$B_Adc] Eql 0     ! If it should be a UNIBUS                     [13]
;    0423  5                      And .Adapter_Type [0] Eql 5          ! .. and string has the right length for 'DW7%0'  [13]
;    0424  5                      And .(Adapter_Type [1])<0, 24> Eql %Asci'DW7' ! .. and it starts out right          [13]
;    0425  5                      And .Adapter_Type [5] Eql %C'0'      ! .. and it ends right                         [13]
;    0426  5                     Then                                  !                                              [13]
;    0427  5                         Found_Adapter = True              ! .. then mark that it's Ok                    [13]
;    0428  5                     Else If .Support_Row [Spt$B_Adc] Eql 32 ! .. if it should be MASSbus                 [13]
;    0429  5                      And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'RH7%0'       [13]
;    0430  5                      And .(Adapter_Type [1])<0, 24> Eql %Asci'RH7' ! .. and it starts out right         [13]
;    0431  5                      And .Adapter_Type [5] Eql %C'0'      ! .. and it ends right                         [13]
;    0432  5                     Then                                  !                                              [13]
```

```
;  0433  5                      Found_Adapter = True                    ! .. then mark it as OK                                  [13]
;  0434  5
;  0435  5                  ELSE If .Support_Row [Spt$B_Adc] Eql 0      ! .. if it should be CIbus                               [18]
;  0436  5                   And .Adapter_Type [0] Eql 5                ! .. and it has right length for 'CI7%0'                 [18]
;  0437  5                   And .(Adapter_Type [1])<0, 24> Eql %Asci'CI7' ! .. and it starts out right                          [18]
;  0438  5                   And .Adapter_Type [5] Eq: %C'0'            ! .. and it ends right                                   [18]
;  0439  5                  Then                                        !                                                        [18]
;  0440  5                      Found_Adapter = True                    ! .. then mark it as OK                                  [18]
;  0441  5
;  0442  5                  End                                         !                                                        [13]
;  0443  4              Else                                            !                                                        [13]
;  0444  4                  Found_Adapter = True;                       ! Pretend we found it, if there shouldn't be any         [13]
;  0445  4
;  0446  4              If .Support_Row [Spt$V_Controller]              ! If there should be a controller, check drives          [13]
;  0447  4              Then                                            !                                                        [13]
;  0448  5                  Begin                                       !                                                        [13]
;  0449  5
;  0450  5                  Local                                       !                                                        [13]
;  0451  5                      Index : Initial (0);                    ! Index in list of drive names                           [13]
;  0452  5
;  0453  5                  Bind                                        ! Bind to vector of names                                [13]
;  0454  5                      Drive_Vector = .Support_Row [Spt$L_Drives] : Vector;        !                                    [13]
;  0455  5
;  0456  5                  If Drive_Vector Eql 0                       ! If no support list                                     [13]
;  0457  5                  Then                                        !                                                        [13]
;  0458  5                      Return DS$_ProgErr;                     ! Return with an error                                   [13]
;  0459  5
;  0460  5                  While Not .Found_Unit And .Drive_Vector [.Index] Neq 0 Do       ! Loop through drive list        [13]
;  0461  6                      Begin                                   !                                                        [13]
;  0462  6
;  0463  6                  Bind                                        ! Map to ASCIC string                                    [13]
;  0464  6                      Unit_Name = .Drive_Vector [.Index] : Vector [, Byte],   ! Map string to check                    [13]
;  0465  6                      Device_Name = Device [Hp$T_Type] : Vector [, Byte];     ! Map device Ptable type              [13]
;  0466  6
;  0467  6                  If .Unit_Name [0] Eql .Device_Name [0]      ! If sizes are equal                                     [13]
;  0468  6                   And Ch$Eql (                               ! .. and names themselves match OK                       [13]
;  0469  6                      .Unit_Name [0],                         !                                                        [13]
;  0470  6                      Unit_Name [1],                          !                                                        [13]
;  0471  6                      .Unit_Name [0],                         !                                                        [13]
;  0472  6                      Device_Name [1]                         !                                                        [13]
;  0473  6                      )                                       !                                                        [13]
;  0474  6                  Then                                        !                                                        [13]
;  0475  7                      Begin                                   !                                                        [13]
;  0476  7                      Found_Unit = True;                      ! OK, this drive is supported                            [13]
;  0477  7                      ExitLoop                                ! Don't check the others                                 [13]
;  0478  6                      End;                                    !                                                        [13]
;  0479  6
;  0480  6                  Index = .Index + 1                          ! Increment table index                                  [13]
;  0481  6                  End                                         ! End of WHILE loop                                      [13]
;  0482  6
;  0483  5                  End                                         !                                                        [13]
;  0484  4              Else                                            !                                                        [13]
;  0485  4                  Found_Unit = True;                          ! Mark as OK if shouldn't be a drive                     [13]
;  0486  4
;  0487  4              Found_Device = (.Found_Adapter And .Found_Unit);     ! Check both search bits                            [13]
;  0488  4
;  0489  4              If .Found_Device
```

```
;  0490  5                 THEN IF ((.SUPPORT_ROW [SPT$B_BTD] NEQ BTD$K_CONSOLE) OR                ! Not console or        [23]
;  0491  6                         ((.SUPPORT_ROW [SPT$B_BTD] EQL BTD$K_CONSOLE) AND               ! (console and          [23]
;  0492  6                          (.DSA$GL_SID [PR$V_SID_TYPE] NEQ PR$_SID_TYPZZZ) AND           !  not ZZZ cpu and      [23]
;  0493  5                          (.SUPPORT_ROW [SPT$B_FILE] EQL 3))          OR                 !  RT11)                [23]
;  0494  6                         ((.SUPPORT_ROW [SPT$B_BTD] EQL BTD$K_CONSOLE) AND               ! (console and          [23]
;  0495  6                          (.DSA$GL_SID [PR$V_SID_TYPE] EQL PR$_SID_TYPZZZ) AND           !  ZZZ cpu and          [23]
;  0496  5                          (.SUPPORT_ROW [SPT$B_FILE] EQL 2)))                            !  ODS)                 [23]
;  0497  4                       THEN EXITLOOP
;  0498  4
;  0499  3                 End;                                                                    !                       [13]
;  0500  3
;  0501  3             Support_Row = .Support_Row + Spt$K_Row_Size     ! Advance to next row       [13]
;  0502  2             End;                                            !                           [13]
;  0503  2
;  0504  2         If Not .Found_Device Then Return Ds$_Error;         ! Return with error, if not found   [13]
;  0505  2
;  0506  2         If Not NullParameter (2)                            ! If output params specified, copy values   [13]
;  0507  2         Then                                                !                           [13]
;  0508  2             .Adc = .Support_Row [Spt$B_Adc];                ! .. copy adapter code      [13]
;  0509  2
;  0510  2         If Not NullParameter (3)                            !                           [13]
;  0511  2         Then                                                !                           [13]
;  0512  2             .Btd = .Support_Row [Spt$B_Btd];                ! .. copy device code       [13]
;  0513  2
;  0514  2         If Not NullParameter (4)                            !                           [13]
;  0515  2         Then                                                !                           [13]
;  0516  2             .File = .Support_Row [Spt$B_File];              ! .. copy file code         [13]
;  0517  2
;  0518  2         If Not NullParameter (5)                            !                           [13]
;  0519  2         Then                                                !                           [13]
;  0520  2             .Init = .Support_Row [Spt$L_Init];              ! .. copy init address      [13]
;  0521  2
;  0522  2         SS$_Normal                                          ! Success                   [13]
;  0523  1         End;                                                ! End of routine            [13]


                                    0FFC 00000          .ENTRY  DSX$CHECK_SUPPORT, Save R2,R3,R4,R5,R6,R7,- ; 0344
                                                                R8,R9,R10,R11                               ;
                5E                  04   C2 00002          SUBL2   #4, SP                                  ;
                                    5B   94 00005          CLRB    FOUND_DEVICE                            ; 0345
                52          04      AC   D0 00007          MOVL    DEVICE, PREV_CONTROLLER                 ; 0361
                53                  52   D0 0000B          MOVL    PREV_CONTROLLER, PREV_ADAPTER           ;
                50                  52   D0 0000E          MOVL    PREV_CONTROLLER, CONTROLLER             ;
                57                  52   D0 00011          MOVL    PREV_CONTROLLER, ADAPTER                ;
                            20      A7   D5 00014 1$:       TSTL    32(ADAPTER)                             ; 0363
                                    0F   13 00017          BEQL    2$                                      ;
                52                  50   D0 00019          MOVL    CONTROLLER, PREV_CONTROLLER             ; 0365
                50                  57   D0 0001C          MOVL    ADAPTER, CONTROLLER                     ; 0366
                53                  57   D0 0001F          MOVL    ADAPTER, PREV_ADAPTER                   ; 0367
                57          20      A7   D0 00022          MOVL    32(ADAPTER), ADAPTER                    ; 0368
                                    EC   11 00026          BRB     1$                                      ;
                51          26      A7   9E 00028 2$:       MOVAB   38(ADAPTER), R1                         ; 0373
                04                  61   91 0002C          CMPB    (R1), #4                                ; 0378
                                    10   12 0002F          BNEQ    3$                                      ;
```

```
                                41494253  8F        01    A1  D1 00031          CMPL    1(R1), #1095320147                          ; 0379
                                                      06  12 00039          BNEQ    3$                                          ;
                                          50          52  D0 0003B          MOVL    PREV_CONTROLLER, CONTROLLER                 ; 0382
                                          57          53  D0 0003E          MOVL    PREV_ADAPTER, ADAPTER                       ; 0383
                                          54       26 A0  9E 00041 3$:       MOVAB   38(R0), CHECK_DEVICE                        ; 0386
                                          55 00C00000' EF  9E 00045          MOVAB   DS$GA_SUPPORT_TABLE, SUPPORT_ROW            ; 0387
                                                      6E  D4 0004C          CLRL    I                                           ; 0401
                                          50          65  D0 0004E 4$:       MOVL    (SUPPORT_ROW), R0                           ; 0399
                                          64          60  91 00051          CMPB    (R0), (CHECK_DEVICE)                        ; 0401
                                                      0E  12 00054          BNEQ    5$                                          ;
                                          52          60  9A 00056          MOVZBL  (R0), R2                                    ; 0403
                                          51          60  9A 00059          MOVZBL  (R0), R1                                    ; 0405
                51              00        01  A0             52  2D 0005C          CMPC5   R2, 1(R0), #0, R1, 1(CHECK_DEVICE)          ; 0406
                                                   01 A4       00062                                                              ;
                                                      03  13 00064 5$:       BEQL    6$                                          ;
                                                   00E2  31 00066          BRW     17$                                         ;
                                                      59  94 00069 6$:       CLRB    FOUND_ADAPTER                               ; 0409
                                                      5A  94 0006B          CLRB    FOUND_UNIT                                  ;
                                          58       07 A5  E9 0006D          BLBC    7(SUPPORT_ROW), 9$                          ; 0415
                                          50       26 A7  9E 00071          MOVAB   38(ADAPTER), R0                             ; 0420
                                                      51  D4 00075          CLRL    R1                                          ; 0422
                                                   04 A5  95 00077          TSTB    4(SUPPORT_ROW)                              ;
                                                      19  12 0007A          BNEQ    7$                                          ;
                                                      51  D6 0007C          INCL    R1                                          ;
                                          05          60  91 0007E          CMPB    (R0), #5                                    ; 0423
                                                      12  12 00081          BNEQ    7$                                          ;
00375744        8F        01    A0          18          00  ED 00083          CMPZV   #0, #24, 1(R0), #3626820                    ; 0424
                                                      06  12 0008D          BNEQ    7$                                          ;
                                          30       05 A0  91 0008F          CMPB    5(R0), #48                                  ; 0425
                                                      37  13 00093          BEQL    9$                                          ;
                                          20       04 A5  91 00095 7$:       CMPB    4(SUPPORT_ROW), #32                         ; 0428
                                                      17  12 00099          BNEQ    8$                                          ;
                                          05          60  91 0009B          CMPB    (R0), #5                                    ; 0429
                                                      12  12 0009E          BNEQ    8$                                          ;
00374852        8F        01    A0          18          00  ED 000A0          CMPZV   #0, #24, 1(R0), #3622994                    ; 0430
                                                      06  12 000AA          BNEQ    8$                                          ;
                                          30       05 A0  91 000AC          CMPB    5(R0), #48                                  ; 0431
                                                      1A  13 000B0          BEQL    9$                                          ;
                                          1A          51  E9 000B2 8$:       BLBC    R1, 10$                                     ; 0435
                                          05          60  91 000B5          CMPB    (R0), #5                                    ; 0436
                                                      15  12 000B8          BNEQ    10$                                         ;
00374943        8F        01    A0          18          00  ED 000BA          CMPZV   #0, #24, 1(R0), #3623235                    ; 0437
                                                      09  12 000C4          BNEQ    10$                                         ;
                                          30       05 A0  91 000C6          CMPB    5(R0), #48                                  ; 0438
                                                      03  12 000CA          BNEQ    10$                                         ;
                                          59          01  90 000CC 9$:       MOVB    #1, FOUND_ADAPTER                           ; 0444
                        3E        07 A5             01  E1 000CF 10$:      BBC     #1, 7(SUPPORT_ROW), 14$                     ; 0446
                                                      56  D4 000D4          CLRL    INDEX                                       ; 0448
                                                   0C A5  D5 000D6          TSTL    12(SUPPORT_ROW)                             ; 0456
                                                      08  12 000D9          BNEQ    11$                                         ;
                                          00 00660020 8F  D0 000DB          MOVL    #6684704, R0                                ; 0458
                                                      04  00 000E2          RET                                                 ;
                                          50       04 AC  D0 000E3 11$:      MOVL    DEVICE, R0                                  ; 0465
                                          58       26 A0  9E 000E7          MOVAB   38(R0), R8                                  ;
                                          27          5A  E8 000EA 12$:      BLBS    FOUND_UNIT, 15$                             ; 0460
                                                   0C B546  D5 000EE          TSTL    @12(SUPPORT_ROW)[INDEX]                     ;
                                                      21  13 000F2          BEQL    15$                                         ;
```

```
                                50        0C B546   D0 000F4          MOVL    @12(SUPPORT_ROW)[INDEX], R0                     ; 0464
                                68              60  91 000F9          CMPC    (R0), (R8)                                      ; 0467
                                                10  12 000FC          BNEQ    13$                                             ;
                                52              60  9A 000FE          MOVZBL  (R0), R2                                        ; 0469
                                51              60  9A 00101          MOVZBL  (R0), R1                                        ; 0471
                51              00        01    A0       52  2D 00104          CMPC5   R2, 1(R0), #0, R1, 1(R8)                ; 0472
                                                      01    A8    0010A                                                       ;
                                                04  13 0010C          BEQL    14$                                             ;
                                                56  D6 0010E 13$:     INCL    INDEX                                           ; 0480
                                                D9  11 00110          BRB     12$                                             ;
                                5A                    01  90 00112 14$:     MOVB    #1, FOUND_UNIT                          ; 0485
                                50                    5A  92 00115 15$:     MCOMB   FOUND_UNIT, R0                          ; 0487
                        5B      59                    50  8B 00118          BICB3   R0, FOUND_ADAPTER, FOUND_DEVICE         ;
                                2C                    5B  E9 0011C          BLBC    FOUND_DEVICE, 17$                       ; 0489
                                        40  8F        05  A5 91 0011F          CMPB    5(SUPPORT_ROW), #64                     ; 0490
                                                      2E  12 00124          BNEQ    18$                                     ;
                                        C5 0000FE17   9F  91 00126          CMPB    @#^X0000FE17, #5                        ; 0492
                                                      06  13 0012D          BEQL    16$                                     ;
                                        03        06  A5 91 0012F          CMPB    6(SUPPORT_ROW), #3                      ; 0493
                                                      1F  13 00133          BEQL    18$                                     ;
                                        40  8F        05  A5 91 00135 16$:     CMPB    5(SUPPORT_ROW), #64                     ; 0494
                                                      0F  12 0013A          BNEQ    17$                                     ;
                                        05 0000FE17   9F  91 0013C          CMPB    @#^X0000FE17, #5                        ; 0495
                                                      06  12 00143          BNEQ    17$                                     ;
                                        02        06  A5 91 00145          CMPB    6(SUPPORT_ROW), #2                      ; 0496
                                                      09  13 00149          BEQL    18$                                     ;
                                55                    10  C0 0014B 17$:     ADDL2   #16, SUPPORT_ROW                        ; 0501
        FEFA            6E                    01        16  F1 0014E          ACBL    #22, #1, I, 4$                          ;
                                08                    5B  E8 00154 18$:     BLBS    FOUND_DEVICE, 19$                       ; 0504
                                50 00660002   8F        D0 00157          MOVL    #6684674, R0                            ;
                                                      04  0015E          RET                                             ;
                                02                    6C  91 0015F 19$:     CMPB    (AP), #2                                ; 0506
                                                      0A  1F 00162          BLSSU   20$                                     ;
                                              08      AC  D5 00164          TSTL    8(AP)                                   ;
                                                      05  13 00167          BEQL    20$                                     ;
                        08      BC            04      A5  9A 00169          MOVZBL  4(SUPPORT_ROW), @ADC                    ; 0508
                                03                    6C  91 0016E 20$:     CMPB    (AP), #3                                ; 0510
                                                      0A  1F 00171          BLSSU   21$                                     ;
                                              0C      AC  D5 00173          TSTL    12(AP)                                  ;
                                                      05  13 00176          BEQL    21$                                     ;
                        0C      BC            05      A5  9A 00178          MOVZBL  5(SUPPORT_ROW), @BTD                    ; 0512
                                04                    6C  91 0017D 21$:     CMPB    (AP), #4                                ; 0514
                                                      0A  1F 00180          BLSSU   22$                                     ;
                                              10      AC  D5 00182          TSTL    16(AP)                                  ;
                                                      05  13 00185          BEQL    22$                                     ;
                        10      BC            06      A5  9A 00187          MOVZBL  6(SUPPORT_ROW), @FILE                   ; 0516
                                05                    6C  91 0018C 22$:     CMPB    (AP), #5                                ; 0518
                                                      0A  1F 0018F          BLSSU   23$                                     ;
                                              14      AC  D5 00191          TSTL    20(AP)                                  ;
                                                      05  13 00194          BEQL    23$                                     ;
                        14      BC            08      A5  D0 00196          MOVL    8(SUPPORT_ROW), @INIT                   ; 0520
                                50                    01  D0 0019B 23$:     MOVL    #1, R0                                  ; 0523
                                                      04  0019E          RET                                             ;

; Routine Size:  415 bytes,    Routine Base:  CODE + 008C
```

```
;    0524  1     %sbttl 'load error status'
;    0525  1
;    0526  1     !++
;    0527  1     ! Functional description:
;    0528  1     !       Set status in FAB field STS and clean up static storage for
;    0529  1     !       return.
;    0530  1     !
;    0531  1     ! Inputs:
;    0532  1     !       R0      error code
;    0533  1     ! Implicit inputs:
;    0534  1     !       R11     Pointer to FAB
;    0535  1     !       also some CBLOCK fields if the CBLOCK has been allocated
;    0536  1     ! Outputs:
;    0537  1     !       none
;    0538  1     ! Implicit outputs:
;    0539  1     !       loads error code into FAB [fab$l_sts]
;    0540  1     ! Side effects:
;    0541  1     !       It will deallocate all nonpaged pool space allocated and
;    0542  1     !       clear the RMS$FILE_TABLE entry used to point to it, as well
;    0543  1     !       as the FAB's IFI which is an index into this table.
;    0544  1     ! Return codes: R0 unaffected
;    0545  1     !--
;    0546  1
;    0547  1     global routine rms$error (code) : jsb_rms =
;    0548  2         begin
;    0549  2
;    0550  2         external register
;    0551  2             cblock = 9 : ref bblock,
;    0552  2             fab = 11 : ref bblock;
;    0553  2
;    0554  2         bind
;    0555  2             cache = cblock [ctl$l_cache1] : vector; ! Make it easy to look at
;    0556  2
;    0557  2         fab [fab$l_sts] = .code;                    ! Load the return code
;    0558  2
;    0559  2         if .cblock eql 0 then return .code;         ! Can't go further if no cblock
;    0560  2
;    0561  2         if not .code                                ! If it was an error
;    0562  2             or .cblock [ctl$b_op] eql ctl$c_close   !   or this is a close
;    0563  2         then
;    0564  3             begin                                   ! Deallocate the world!!!!!
;    0565  3
;    0566  3             if .cblock [ctl$b_op] eql ctl$c_open or .cblock [ctl$b_op] eql ctl$c_close
;    0567  3                                                     ! If we should deallocate
;    0568  3             then
;    0569  4                 begin
;    0570  4
;    0571  4                 if .cblock [ctl$l_filhdr] neq 0     ! If file header space was allocated
;    0572  4                 then
;    0573  5                     begin
;    0574  5                     (.cblock [ctl$l_filhdr] + 8)<0, 16> = ctl$c_filhdr_siz; ! Set length of filhdr
;    0575  5                     exe$deanonpaged (.cblock [ctl$l_filhdr])        ! Deallocate it
;    0576  4                     end;
;    0577  4
;    0578  4                 if .cblock [ctl$l_rpb] neq 0        ! If we allocated an RPB
;    0579  4                 then
;    0580  4                     Dsr$Deallocate_Pseudo_RPB (.cblock [ctl$l_rpb]);        ! Deallocate it                [10]
```

```
;    0581  4
;    0582  4                 if .cblock [ctl$w_file_len] neq 0   ! If we allocated a string
;    0583  4                 then
;    0584  5                     begin                          ! De-allocate it
;    0585  5                     (.cblock [ctl$l_file_ptr] + 8)<0, 16> = .cblock [ctl$w_file_alloc];
;    0586  5                                                    ! Load IRP$W_SIZE field
;    0587  5                     exe$deanonpaged (.cblock [ctl$l_file_ptr])      ! De-allocate the string
;    0588  4                     end;
;    0589  4
;    0590  4         !
;    0591  4         ! Now deallocate cache buffers if any were allocated
;    0592  4         !
;    0593  4
;    0594  4                 incr i from 0 to 2 do
;    0595  4
;    0596  4                     if .cache [.i] neq 0           ! If this buffer was allocated
;    0597  4                     then
;    0598  5                         begin
;    0599  5                         (.cache [.i] + 8)<0, 16> = .cblock [ctl$w_cache_size]*512;  ! Set size of buffer
;    0600  5                         exe$deanonpaged (.cache [.i])       ! Deallocate it
;    0601  4                         end;
;    0602  4
;    0603  4                 exe$deanonpaged (.cblock);         ! De-allocate the block
;    0604  4                 rms$file_table [.fab [fab$w_ifi]] = 0;      ! Clear table entry
;    0605  4                 cblock = 0;                        ! Clear cblock pointer
;    0606  4                 fab [fab$w_ifi] = 0                ! Clear IFI
;    0607  4                 end
;    0608  4
;    0609  2             end;
;    0610  2
;    0611  2         if .cblock neq 0 then cblock [ctl$b_op] = 0;        ! Clear operation code
;    0612  2
;    0613  2         .code                                      ! Return what we came with
;    0614  1         end;                                       ! of rms$error


                                    3E   BB 00000 RMS$ERROR::
                                                  PUSHR   #^M<R1,R2,R3,R4,R5>                 ; 0547
                              55    50   D0 00002           MOVL    R0, R5                      ;
                        08    AB    55   D0 00005           MOVL    CODE, 8(FAB)                ; 0557
                                    59   D5 00009           TSTL    CBLOCK                      ; 0559
                                    7F   13 0000B           BEQL    9$                          ;
                              05    55   E9 0000D           BLBC    CODE, 1$                    ; 0561
                              04    69   91 00010           CMPB    (CBLOCK), #4                ; 0562
                                    71   12 00013           BNEQ    8$                          ;
                              01    69   91 00015 1$:       CMPB    (CBLOCK), #1                ; 0566
                                    05   13 00018           BEQL    2$                          ;
                              04    69   91 0001A           CMPB    (CBLOCK), #4                ;
                                    67   12 0001D           BNEQ    8$                          ;
                              50          54   A9 D0 0001F 2$:       MOVL    84(CBLOCK), R0      ; 0571
                                    0C   13 00023           BEQL    3$                          ;
                        08    A0    0200   8F B0 00025           MOVW    #512, 8(R0)             ; 0574
                                00000000G  EF 16 0002B           JSB     EXE$DEANONPAGED         ; 0575
                                      38   A9 D5 00031 3$:       TSTL    56(CBLOCK)              ; 0578
```

```
                                                08  13 00034          BEQL    4$                                          ;
                                             38 A9  DD 00036          PUSHL   56(CBLOCK)                                  ; 0580
                                      FE10   CF     01  FB 00039          CALLS   #1, DSR$DEALLOCATE_PSEUDO_RPB               ;
                                             48 A9  B5  0003E 4$:       TSTW    72(CBLOCK)                                  ; 0582
                                                0F  13  00041          BEQL    5$                                          ;
                                          50    4C A9  D0  00043          MOVL    76(CBLOCK), R0                              ; 0585
                                       08 A0    4A A9  B0  00047          MOVW    74(CBLOCK), 8(R0)                           ;
                                          00000000G   EF  16  0004C          JSB     EXE$DEANONPAGED                             ; 0587
                                                54    D4  00052 5$:       CLRL    I                                           ; 0594
                                          50    2C A944 D0  00054 6$:       MOVL    44(CBLOCK)[I], R0                           ; 0596
                                                0E  13  00059          BEQL    7$                                          ;
                                08 A0    0A A9    0200  8F  A5  0005B          MULW3   #512, 10(CBLOCK), 8(R0)                     ; 0599
                                          00000000G   EF  16  00063          JSB     EXE$DEANONPAGED                             ; 0600
                                       E7           54  02  F3  00069 7$:       AOBLEQ  #2, I, 6$                           ; 0596
                                          50        59  D0  0006D          MOVL    CBLOCK, R0                                  ; 0603
                                          00000000G   EF  16  00070          JSB     EXE$DEANONPAGED                             ;
                                          50    02  AB  3C  00076          MOVZWL  2(FAB), R0                                  ; 0604
                                          00000000'EF40   D4  0007A          CLRL    RMS$FILE_TABLE[R0]                          ;
                                                59  D4  00081          CLRL    CBLOCK                                      ; 0605
                                             02 AB  B4  00083          CLRW    2(FAB)                                      ; 0606
                                                59  D5  00086 8$:       TSTL    CBLOCK                                      ; 0611
                                                02  13  00088          BEQL    9$                                          ;
                                                69  94  0008A          CLRB    (CBLOCK)                                    ;
                                          50    55  D0  0008C 9$:       MOVL    CODE, R0                                    ; 0614
                                                3E  BA  0008F          POPR    #^M<R1,R2,R3,R4,R5>                         ;
                                                    05  00091          RSB                                                 ;

; Routine Size:  146 bytes,    Routine Base:  CODE + 022B

;  0615  1
```

```
;    0616  1      %sbttl 'load XAB'
;    0617  1
;    0618  1      global routine rms$load_xab : call_rms novalue =
;    0619  1
;    0620  1      !++
;    0621  1      ! Functional description:
;    0622  1      !       set the fields of the FHC XAB.
;    0623  1      !
;    0624  1      ! Inputs:
;    0625  1      !       none
;    0626  1      !
;    0627  1      ! Implicit inputs:
;    0628  1      !       R9              Pointer to the RMS control block
;    0629  1      !       R11             Pointer to the FAB block
;    0630  1      !
;    0631  1      !       all other fields are taken from the FAB or the RMS internal
;    0632  1      !       control block.
;    0633  1      !
;    0634  1      ! Outputs:
;    0635  1      !       none
;    0636  1      !
;    0637  1      ! Implicit Outputs:
;    0638  1      !       if the XAB pointer of the FAB is not zero, the XAB block is
;    0639  1      !       filled in.
;    0640  1      !
;    0641  1      ! Side Effects:
;    0642  1      !       none
;    0643  1      !
;    0644  1      ! Return codes:
;    0645  1      !       none
;    0646  1      !--
;    0647  1
;    0648  2          begin
;    0649  2
;    0650  2          external register
;    0651  2              cblock = 9 : ref bblock,                ! Pointer to control block
;    0652  2              fab = 11 : ref bblock;                  ! Pointer to FAB
;    0653  2
;    0654  2          local
;    0655  2              xab : ref bblock;
;    0656  2
;    0657  2          xab = .fab [fab$l_xab];                     ! Get first XAB address
;    0658  2
;    0659  2          while .xab neq 0                            ! while there are more XABs
;    0660  2              and .xab [xab$b_cod] neq xab$c_fhc do   !  and it's not a FHCXAB
;    0661  2              xab = .xab [xab$l_nxt];                 ! link to next XAB in chain
;    0662  2
;    0663  2          if .xab eql 0                               ! If we ran out of XABs
;    0664  2              or .xab [xab$b_bln] neq xab$c_fhclen    !  or this isn't really an FHC
;    0665  2          then
;    0666  2              return;                                 ! return to caller
;    0667  2
;    0668  2          xab [xab$b_atr] = .fab [fab$b_rat];         ! Copy attributes
;    0669  2          xab [xab$l_ebk] = .cblock [ctl$l_eofvbn];   ! Copy end-of-file VBN
;    0670  2          xab [xab$w_ffb] = .cblock [ctl$w_offbyte];  ! Copy end-of-file byte number
;    0671  2          xab [xab$b_hsz] = .fab [fab$b_fsz];         ! Copy fixed header size
;    0672  2          xab [xab$w_lrl] =                           ! Longest is maximum
```

```
;    0673  2          xab [xab$w_mrz] = .fab [fab$w_mrs];          ! Copy max record length
;    0674  2          (xab [xab$b_rfo])<0, 4> = .fab [fab$b_rfm]; ! Copy record format
;    0675  2          (xab [xab$b_rfo])<4, 4> = fab$c_seq;        ! Always sequential
;    0676  2          xab [xab$l_sbn] = .cblock [ctl$l_startlbn]  ! Copy starting block number
;    0677  1          end;                                        ! of rms$load_xab


                                                 0000 00000          .ENTRY  RMS$LOAD_XAB, Save nothing                      ; 0618
                                  50      24  AB D0 00002          MOVL    36(FAB), XAB                                    ; 0657
                                              0B 13 00006 1$:     BEQL    2$                                              ; 0659
                                  1D          60 91 00008         CMPB    (XAB), #29                                      ; 0660
                                              06 13 0000B         BEQL    2$                                              ;
                                  50      04  A0 D0 0000D         MOVL    4(XAB), XAB                                     ; 0661
                                              F3 11 00011         BRB     1$                                              ;
                                              50 D5 00013 2$:     TSTL    XAB                                             ; 0663
                                              37 13 00015         BEQL    3$                                              ;
                                  2C      01  A0 91 00017         CMPB    1(XAB), #44                                     ; 0664
                                              31 12 0001B         BNEQ    3$                                              ;
                              09  A0      1E  AB 90 0001D         MOVB    30(FAB), 9(XAB)                                 ; 0668
                              10  A0      44  A9 D0 00022         MOVL    68(CBLOCK), 16(XAB)                             ; 0669
                              14  A0      42  A9 B0 00027         MOVW    66(CBLOCK), 20(XAB)                             ; 0670
                              17  A0      3F  AB 90 0002C         MOVB    63(FAB), 23(XAB)                                ; 0671
                                  51      36  AB 3C 00031         MOVZWL  54(FAB), R1                                     ; 0673
                              18  A0          51 B0 00035         MOVW    R1, 24(XAB)                                     ;
                              0A  A0          51 B0 00039         MOVW    R1, 10(XAB)                                     ;
    08    A0                      00      1F  AB F0 0003D         INSV    31(FAB), #0, #4, 8(XAB)                         ; 0674
                              08  A0      F0  8F 8A 00044         BICB2   #240, 8(XAB)                                    ; 0675
                              28  A0      50  A9 D0 00049         MOVL    80(CBLOCK), 40(XAB)                             ; 0676
                                                 04 0004E 3$:     RET                                                     ; 0677

; Routine Size:  79 bytes,    Routine Base:  CODE + 028D

;    0678  1
```

```
;    0679  1     %sbttl 'allocate cache blocks'
;    0680  1
;    0681  1     global routine rms$alloc_cache (bufsiz) : jsb_cblock =
;    0682  1
;    0683  1     !++
;    0684  1     ! Functional description:
;    0685  1     !       Allocate cache blocks.  Since Magtape requires larger cache
;    0686  1     !       buffers than other devices, this routine allows each format-
;    0687  1     !       specific OPEN routine to allocate whatever buffer size is
;    0688  1     !       necessary.
;    0689  1     !
;    0690  1     ! Inputs:
;    0691  1     !       bufsiz          size of cache buffer needed in pages
;    0692  1     !
;    0693  1     ! Implicit inputs:
;    0694  1     !       R9              pointer to control block
;    0695  1     !
;    0696  1     ! Outputs:
;    0697  1     !       none
;    0698  1     !
;    0699  1     ! Implicit outputs:
;    0700  1     !       none
;    0701  1     !
;    0702  1     ! Side effects:
;    0703  1     !       'non-paged' pool space is allocated for each cache buffer (3).
;    0704  1     !       The cache buffer pointers in the control block are set to point
;    0705  1     !       to them, and the CACHE_SIZE word is set to the buffer length
;    0706  1     !       in pages
;    0707  1     !
;    0708  1     ! Return codes
;    0709  1     !       RMS$_NORMAL             if everything was OK
;    0710  1     !       RMS$_DME                if couldn't allocate memory
;    0711  1     !-
;    0712  2         begin
;    0713  2
;    0714  2         external register
;    0715  2             cblock = 9 : ref bblock;                ! Map to control block
;    0716  2
;    0717  2         Local                                       !                       [10]
;    0718  2             Size;                                   !                       [10]
;    0719  2
;    0720  2         cblock [ctl$w_cache_size] = size = .bufsiz; ! Set buffer size to alloc.
;    0721  2         size = .size*512;                           ! Convert to bytes
;    0722  2
;    0723  2         If Not Exe$Alononpaged (.Size;, CBlock [Ctl$L_Cache1])      !       [10]
;    0724  2         Then                                        !               [10]
;    0725  2             Return Rms$_Dme;        ! Allocate A Buffer             [10]
;    0726  2
;    0727  2         If Not Exe$Alononpaged (.Size;, CBlock [Ctl$L_Cache2])      !       [10]
;    0728  2         Then                                        !               [10]
;    0729  2             Return Rms$_Dme;                        !               [10]
;    0730  2
;    0731  2         If Not Exe$Alononpaged (.Size;, CBlock [Ctl$L_Cache3])      !       [10]
;    0732  2         Then                                        !               [10]
;    0733  2             Return Rms$_Dme;                        !               [10]
;    0734  2
;    0735  2         rms$_normal
```

```
:   0736  1          end;                                ! of rms$alloc_cache

                                                1C  BB 00000 RMS$ALLOC_CACHE::
                                                             PUSHR   #^M<R2,R3,R4>                                   ; 0681
                          54                    50  D0 00002          MOVL    BUFSIZ, SIZE                          ; 0720
                      0A  A9                    50  B0 00005          MOVW    BUFSIZ, 10(CBLOCK)                    ;
                  54      54                    09  78 00009          ASHL    #9, SIZE, SIZE                        ; 0721
                          51                    54  D0 0000D          MOVL    SIZE, R1                              ; 0723
                              00000000G         EF  16 00010          JSB     EXE$ALONONPAGED                       ;
                      2C  A9                    52  D0 00016          MOVL    R2, 44(CBLOCK)                        ;
                          20                    50  E9 0001A          BLBC    R0, 1$                                ;
                          51                    54  D0 0001D          MOVL    SIZE, R1                              ; 0727
                              00000000G         EF  16 00020          JSB     EXE$ALONONPAGED                       ;
                      30  A9                    52  D0 00026          MOVL    R2, 48(CBLOCK)                        ;
                          10                    50  E9 0002A          BLBC    R0, 1$                                ;
                          51                    54  D0 0002D          MOVL    SIZE, R1                              ; 0731
                              00000000G         EF  16 00030          JSB     EXE$ALONONPAGED                       ;
                      34  A9                    52  D0 00036          MOVL    R2, 52(CBLOCK)                        ;
                          09                    50  E8 0003A          BLBS    R0, 2$                                ;
                          50  000184D4          8F  D0 0003D 1$:      MOVL    #99540, R0                            ; 0733
                                                07  11 00044          BRB     3$                                    ;
                          50  00010001          8F  D0 00046 2$:      MOVL    #65537, R0                            ; 0736
                                                1C  BA 0004D 3$:      POPR    #^M<R2,R3,R4>                         ;
                                                05  0004F            RSB                                           ;

; Routine Size:  80 bytes,    Routine Base:  CODE + 030C

;   0737  1
```

```
;    0738  1     %sbttl 'initialize static table'
;    0739  1
;    0740  1     !++
;    0741  1     ! Functional description:
;    0742  1     !       Set the file table pointer locations to 0.
;    0743  1     !
;    0744  1     ! Inputs: none
;    0745  1     ! Implicit inputs: none
;    0746  1     ! Outputs: none
;    0747  1     ! Implicit outputs: none
;    0748  1     ! Side effects: none
;    0749  1     ! Status return: none
;    0750  1     !--
;    0751  1
;    0752  1     global routine rms$init : jsb_none novalue =
;    0753  1
;    0754  1         incr i from 1 to 3 do
;    0755  1             rms$file_table [.i] = 0;

                                50            01  D0 00000 RMS$INIT::
                                                      MOVL    #1, I                           ; 0754
                           00000000'EF40      D4 00003 1$:    CLRL    RMS$FILE_TABLE[I]       ; 0755
                     F5         50            03  F3 0000A          AOBLEQ  #3, I, 1$       ;
                                                   05 0000E          RSB                     ;

; Routine Size:  15 bytes,    Routine Base:  CODE + 035C

;    0756  1
```

```
;    0757  1     %sbttl 'clean up code'
;    0758  1
;    0759  1     global routine rms$cleanup : jsb_none novalue =
;    0760  1
;    0761  1     !++
;    0762  1     ! Functional description:
;    0763  1     !       Called from CLEANUP to de-allocate any RMS static storage from
;    0764  1     !       files not properly closed due to some error.
;    0765  1     !
;    0766  1     ! Inputs : none
;    0767  1     ! Implicit inputs:
;    0768  1     !       rms$file_table  pointers to control blocks
;    0769  1     ! Outputs: None
;    0770  1     ! Implicit outputs: none
;    0771  1     ! Side effects:
;    0772  1     !       deallocates any RMS static storage
;    0773  1     !--
;    0774  1
;    0775  1         incr i from 1 to rms$c_maxfiles do
;    0776  1
;    0777  1             if .rms$file_table [.i] neq 0
;    0778  1             then
;    0779  2                 begin
;    0780  2
;    0781  2                 local
;    0782  2                     x : $fab_decl;                  ! Allocate fake FAB
;    0783  2
;    0784  2                 global register
;    0785  2                     cblock = 9 : ref bblock ,
;    0786  2                     rab = 10,
;    0787  2                     fab = 11 : ref bblock;
;    0788  2
;    0789  2                 cblock = .rms$file_table [.i];      ! Map the control block
;    0790  2                 fab = x;                            ! Map the fake FAB
;    0791  2                 rab = 0;                            ! No RAB
;    0792  2                 fab [fab$w_ifi] = .i;               ! Set the file_table index
;    0793  2                 cblock [ctl$b_op] = ctl$c_close;    ! Force de-allocate
;    0794  2                 rms$error (0)                       ! De-allocate it all
;    0795  1                 end;


                    0E00    8F   BB 00000 RMS$CLEANUP::
                                          PUSHR   #^M<R9,R10,R11>                 ; 0759
                5E    B0    AE   9E 00004         MOVAB   -80(SP), SP             ;
                51          01   D0 00008         MOVL    #1, I                   ; 0775
                50 00000000'EF41   D0 0000B 1$:     MOVL    RMS$FILE_TABLE[I], R0   ; 0777
                            14   13 00013         BEQL    2$                      ;
                59          50   D0 00015         MOVL    R0, CBLOCK              ; 0789
                58          6E   9E 00018         MOVAB   X, FAB                  ; 0790
                            5A   D4 0001B         CLRL    RAB                     ; 0791
            02  AB          51   B0 0001D         MOVW    I, 2(FAB)               ; 0792
                69          04   90 00021         MOVB    #4, (CBLOCK)            ; 0793
                            50   D4 00024         CLRL    R0                      ; 0794
                          FE97   30 00026         BSBW    RMS$ERROR               ;
```

```
                 DE            51        03  F3 00029 2$:    AOBLEQ  #3, I, 1$                          ; 0777
                               5E        50  AE  9E 0002D     MOVAB   8C(SP), SP                        ; 0795
                                     0E00  8F  BA 00031     POPR    #^M<R9,R10,R11>                     ;
                                           05  00035     RSB                                         ;

; Routine Size:  54 bytes,    Routine Base:  CODE + 036B

;   0796  1
```

```
;    0797  1     %sbttl 'load error code in RAB'
;    0798  1     routine rab$error (code) : jsb_rms =
;    0799  1
;    0800  1     !++
;    0801  1     ! Functional description:
;    0802  1     !       Sets error code into RAB
;    0803  1     ! Inputs:
;    0804  1     !       code            the error code
;    0805  1     ! Implicit inputs:
;    0806  1     !       R10             pointer to RAB
;    0807  1     ! Outputs: none
;    0808  1     ! Implicit outputs: none
;    0809  1     ! Side effects: none
;    0810  1     ! Return status:
;    0811  1     !       same as code input
;    0812  1     !--
;    0813  1
;    0814  2         begin
;    0815  2
;    0816  2         external register
;    0817  2             rab = 10 : ref bblock;
;    0818  2
;    0819  3         (rab [rab$l_sts] = .code)
;    0820  1         end;                                        ! of rab$error


                        08  AA          50  D0 00000 RAB$ERROR:
                                                    MOVL    CODE, 8(RAB)                    ; 0819
                                        05  00004    RSB                                     ; 0820

; Routine Size:  5 bytes,    Routine Base:  CODE + 03A1
```

```
;    0821  1     %sbttl 'load FAB error code'
;    0822  1     routine fab$error (code) : jsb_rms =
;    0823  1
;    0824  1     !++
;    0825  1     ! Functional description:
;    0826  1     !       Sets error code into FAB
;    0827  1     ! Inputs:
;    0828  1     !       code            the error code
;    0829  1     ! Implicit inputs:
;    0830  1     !       R11             pointer to FAB
;    0831  1     ! Outputs: none
;    0832  1     ! Implicit outputs: none
;    0833  1     ! Side effects: none
;    0834  1     ! Return status:
;    0835  1     !       same as code input
;    0836  1     !--
;    0837  1
;    0838  2         begin
;    0839  2
;    0840  2         external register
;    0841  2             fab = 11 : ref bblock;
;    0842  2
;    0843  3         (fab [fab$l_sts] = .code)
;    0844  1         end;                                        ! of fab$error


                        08   AB          50  D0 00000 FAB$ERROR:
                                                 MOVL    CODE, 8(FAB)                    ; 0843
                                             05 00004    RSB                             ; 0844

; Routine Size:  5 bytes,    Routine Base:  CODE + 03A6
```

```
;    0845  1     %sbttl 'verify validity of FAB'
;    0846  1     routine verify_fab : jsb_rms =
;    0847  1
;    0848  1     !++
;    0849  1     ! Functional description:
;    0850  1     !       Check the ID fields of the FAB to be sure it's really a FAB
;    0851  1     ! Inputs: none
;    0852  1     ! Implicit inputs:
;    0853  1     !       R11             pointer to the FAB to verify
;    0854  1     ! Outputs: none
;    0855  1     ! Implicit outputs: none
;    0856  1     ! Side effects: none
;    0857  1     ! Return codes:
;    0858  1     !       0               not a FAB
;    0859  1     !       1               A good FAB
;    0860  1     !--
;    0861  1
;    0862  2         begin
;    0863  2
;    0864  2         external register
;    0865  2             fab = 11 : ref bblock;
;    0866  2
;    0867  2         if .fab [fab$b_bid] neq fab$c_bid           ! If the ID byte is wrong,
;    0868  2             or .fab [fab$b_bln] neq fab$c_bln       !  or the length is wrong
;    0869  2         then
;    0870  2             return 0;                               ! it's no good
;    0871  2
;    0872  2         1
;    0873  1         end;                                        ! of verify_fab


                                    03              6B  91 00000 VERIFY_FAB:
                                                                 CMPB    (FAB), #3                       ; 0867
                                                    07  12 00003         BNEQ    1$                      ;
                                50  8F      01      AB  91 00005         CMPB    1(FAB), #80             ; 0868
                                                    03  13 0000A         BEQL    2$                      ;
                                                    50  D4 0000C 1$:     CLRL    R0                      ; 0870
                                                        05 0000E         RSB                             ;
                                    50              01  D0 0000F 2$:     MOVL    #1, R0                  ; 0873
                                                        05 00012         RSB                             ;

; Routine Size:  19 bytes,    Routine Base:  CODE + 03AB
```

```
;    0874  1     %sbttl 'verify cblock'
;    0875  1     routine verify_cblock : jsb_rms =
;    0876  1
;    0877  1     !++
;    0878  1     ! Functional description:
;    0879  1     !       Initialize the global CBLOCK pointer (R9) by using a FAB's
;    0880  1     !       IFI field as index into the RMS$FILE_TABLE array of cblock
;    0881  1     !       pointers.
;    0882  1     ! Inputs: none
;    0883  1     ! Implicit inputs:
;    0884  1     !       R11             pointer to FAB
;    0885  1     ! Outputs: none
;    0886  1     ! Implicit outputs:
;    0887  1     !       R9              pointer to CBLOCK
;    0888  1     ! Side Effects: none
;    0889  1     ! Return codes:
;    0890  1     !       0               IFI or RMS$FILE_TABLE entry is invalid
;    0891  1     !       1               Alright
;    0892  1     !--
;    0893  1
;    0894  2         begin
;    0895  2
;    0896  2         external register
;    0897  2             cblock = 9,                             ! Global pointer to cblock
;    0898  2             fab = 11 : ref bblock;                  ! Global pointer to FAB
;    0899  2
;    0900  2         local
;    0901  2             ifi;
;    0902  2
;    0903  2         ifi = .fab [fab$w_ifi];                     ! Get the IFI index
;    0904  2
;    0905  2         if .ifi leq 0 or .ifi gtr rms$c_maxfiles    ! If the ifi is bad
;    0906  2             or (cblock = .rms$file_table [.ifi]) eql 0      ! or the entry is bad
;    0907  2         then
;    0908  2             return (cblock = 0);                    ! Clear cblock and return
;    0909  2
;    0910  2         1
;    0911  1         end;                                        ! of verify_cblock


                                50       02   AB  3C 00000 VERIFY_CBLOCK:
                                                            MOVZWL  2(FAB), IFI                     ; 0903
                                                 0F  15 00004       BLEQ    1$                      ; 0905
                                03               50  D1 00006       CMPL    IFI, #3                 ;
                                                 0A  14 00009       BGTR    1$                      ;
                                59 00000000'EF   40  D0 0000B       MOVL    RMS$FILE_TABLE[IFI], CBLOCK     ; 0906
                                                 05  12 00013       BNEQ    2$                      ;
                                                 59  D4 00015 1$:    CLRL    CBLOCK                  ; 0908
                                                 50  D4 00017       CLRL    R0                      ;
                                                     05 00019       RSB                             ;
                                50               01  D0 0001A 2$:    MOVL    #1, R0                  ; 0911
                                                     05 0001D       RSB                             ;

; Routine Size:  30 bytes,    Routine Base:  CODE + 03BE
```

```
;    0912  1     %sbttl 'verify validity of RAB'
;    0913  1     routine verify_rab : jsb_rms =
;    0914  1
;    0915  1     !++
;    0916  1     ! Functional description:
;    0917  1     !       Check the ID fields of the RAB to be sure it's really a RAB
;    0918  1     ! Inputs: none
;    0919  1     ! Implicit inputs:
;    0920  1     !       R10             pointer to the RAB to verify
;    0921  1     ! Outputs: none
;    0922  1     ! Implicit outputs: none
;    0923  1     ! Side effects: none
;    0924  1     ! Return codes:
;    0925  1     !       0               not a RAB
;    0926  1     !       1               A good RAB
;    0927  1     !--
;    0928  1
;    0929  2         begin
;    0930  2
;    0931  2         external register
;    0932  2             rab = 10 : ref bblock;
;    0933  2
;    0934  2         if .rab [rab$b_bid] neq rab$c_bid           ! If the ID byte is wrong,
;    0935  2             or .rab [rab$b_bln] neq rab$c_bln       !  or the length is wrong
;    0936  2         then
;    0937  2             return 0;                               ! it's no good
;    0938  2
;    0939  2         1
;    0940  1         end;                                        ! of verify_rab


                                01          6A  91 00000 VERIFY_RAB:
                                                         CMPB    (RAB), #1                       ; 0934
                                            07  12 00003         BNEQ    1$                      ;
                                44  8F      01  AA  91 00005         CMPB    1(RAB), #68             ; 0935
                                            03  13 0000A         BEQL    2$                      ;
                                            50  D4 0000C 1$:     CLRL    R0                      ; 0937
                                            05 0000E             RSB                             ;
                                50          01  D0 0000F 2$:     MOVL    #1, R0                  ; 0940
                                            05 00012             RSB                             ;

; Routine Size:  19 bytes,    Routine Base:  CODE + 03DC
```

```
;    0941  1     %sbttl 'RMS$OPEN routine'
;    0942  1
;    0943  1     global routine rms$open (fabptr) =
;    0944  1
;    0945  1     !++
;    0946  1     ! Functional Description:
;    0947  1     !       Search for the file designated by the FAB filename field (with
;    0948  1     !       appropriate defaults added).  If file is found, set fields in
;    0949  1     !       FAB, NAM and XAB blocks to values retrieved from the file header
;    0950  1     !       or other media areas.
;    0951  1     !
;    0952  1     ! Inputs:
;    0953  1     !       fabptr          Address of the FAB block to control access to
;    0954  1     !                       this file.
;    0955  1     !
;    0956  1     ! Implicit inputs:
;    0957  1     !       fab$l_nam       points to the NAM block associated with this FAB.
;    0958  1     !       fab$l_xab       points to the XAB chain associated with this FAB.
;    0959  1     !       several other pointers with the FAB, NAM and XAB blocks which
;    0960  1     !       determine the type of file manipulations, what values are to be
;    0961  1     !       returned, etc.
;    0962  1     !
;    0963  1     ! Outputs:
;    0964  1     !       R0              status code
;    0965  1     !
;    0966  1     ! Implicit Outputs:
;    0967  1     !       many fields in the FAB, NAM and XAB are set to describe the file
;    0968  1     !       just opened.
;    0969  1     !
;    0970  1     ! Side Effects:
;    0971  1     !       Dynamic pool space is allocated and initialized.  This includes
;    0972  1     !       a copy of the file header, and several blocks for file caching.
;    0973  1     !
;    0974  1     ! Return codes:
;    0975  1     !       RMS$_NORMAL                     Everything's cool
;    0976  1     !       RMS$_ACC                        Error accessing file
;    0977  1     !       RMS$_FAB                        Bad FAB
;    0978  1     !       RMS$_DME                        insufficient memory for internal
;    0979  1     !                                       data structures
;    0980  1     !       RMS$_ORG                        Invalid file organization (only
;    0981  1     !                                       Sequential supported)
;    0982  1     !--
;    0983  1
;    0984  2         begin
;    0985  2
;    0986  2         global register
;    0987  2             cblock = 9 : ref bblock ,               ! Pointer to control block
;    0988  2             rab = 10 : ref bblock ,                 ! RMS package globals
;    0989  2             fab = 11 : ref bblock;
;    0990  2
;    0991  2         local
;    0992  2             CBlockSize : Word,                      ! Return from AloNonPaged
;    0993  2             file_index,                             ! Potential IFI
;    0994  2             filedesc : bblock [dsc$c_s_bln*5],      ! Descriptors for filename parts
;    0995  2             filename : bblock [nam$c_maxrss],       ! Final filename buffer
;    0996  2             filenamedesc : bblock [dsc$c_s_bln];    ! Descriptor for filename
;    0997  2
```

```
;    0998  2      !
;    0999  2      ! first, check consistency.  Specifically, be sure that the
;    1000  2      ! 'FAB' passed really IS a FAB block.  Accept no substitutes
;    1001  2      !
;    1002  2          rab = 0;                                    ! No RAB for OPEN service
;    1003  2          cblock = 0;                                 ! No cblock yet
;    1004  2          fab = .fabptr;                              ! Set FAB pointer
;    1005  2
;    1006  2          if not verify_fab () then return (rms$_fab);        ! Invalid FAB
;    1007  2
;    1008  2          if .fab [fab$b_org] neq fab$c_seq then return rms$error (rms$_org); ! DS only supports sequential
;    1009  2
;    1010  2          fab [fab$w_ifi] = 0;                        ! Pre-clear ifi
;    1011  3          file_index = (incr i from 1 to 3 do         ! Find un-used IFI
;    1012  2              if .rms$file_table [.i] eql 0 then exitloop .i);
;    1013  2
;    1014  2          if .file_index eql -1                       ! If no open slot,
;    1015  2          then
;    1016  2              return (fab [fab$l_sts] = rms$_dme);    ! Error
;    1017  2
;    1018  2          If Not Exe$Alononpaged (Ctl$C_Length; CblockSize, CBlock)   !                                   [11]
;    1019  2          Then                                                                                            [10]
;    1020  2              Return (Fab [Fab$L_Sts] = Rms$_Dme);    ! Error If Can'T Allocate                           [10]
;    1021  2
;    1022  2          Rms$File_Table [.File_Index] = .CBlock;     ! Set file pointer
;    1023  2          ch$fill (0, ctl$c_length, .cblock);         ! Clear the table
;    1024  2          CBlock [Ctl$W_Size] = .CblockSize;          ! set structure size                                [11]
;    1025  2          cblock [ctl$b_op] = ctl$c_open;             ! Set operation code
;    1026  2          fab [fab$w_ifi] = .file_index;              ! Store pointer to it
;    1027  2
;    1028  2      !
;    1029  2      ! Initialize FAB fields
;    1030  2      !
;    1031  2          fab [fab$w_mrs] = fab [fab$b_bks] =         ! Clear a few fields
;    1032  2          fab [fab$l_stv] = fab [fab$l_mrn] = fab [fab$l_sdc] = 0;
;    1033  2          fab [fab$l_sts] = rms$_suc;                 ! Pre-set success code
;    1034  2          ch$fill (0, dsc$c_s_bln*5, filedesc);       ! Initialize filespec parts
;    1035  2          breakup_file (def$c_bkstp_len, def$q_backstop, filedesc);   ! Ultimate system default
;    1036  2          breakup_file (.def$q_dev [dsc$w_length],    ! Load DS default device
;    1037  2              .def$q_dev [dsc$a_pointer], filedesc);
;    1038  2          breakup_file (.def$q_dir [dsc$w_length],    ! Load DS default filespec
;    1039  2              .def$q_dir [dsc$a_pointer], filedesc);
;    1040  2          breakup_file (.fab [fab$b_dns],             ! Load user's default filespec
;    1041  2              .fab [fab$l_dna], filedesc);
;    1042  2          breakup_file (.fab [fab$b_fns],             ! Load filespec
;    1043  2              .fab [fab$l_fna], filedesc);
;    1044  3          begin
;    1045  4          begin
;    1046  4
;    1047  4          local
;    1048  4              pointer,
;    1049  4              length;
;    1050  4
;    1051  4          pointer = filename;                         ! Point to filename buffer
;    1052  4          length = nam$c_maxrss;
;    1053  4          filenamedesc [dsc$a_pointer] = .pointer;    ! And set final descriptor
;    1054  4
```

```
;    1055  4          incr i from 0 to 4 do                          ! For each field
;    1056  5              begin
;    1057  5
;    1058  5              bind
;    1059  5                  d = filedesc + .i*dsc$c_s_bln : bblock;     ! Isolate field to load
;    1060  5
;    1061  5              local
;    1062  5                  len,
;    1063  5                  ptr;
;    1064  5
;    1065  5              len = .d [dsc$w_length];
;    1066  5              ptr = .d [dsc$a_pointer];
;    1067  5
;    1068  5              while (len = .len - 1) geq 0 and (length = .length - 1) geq 0 do
;    1069  5                  ch$wchar_a (ch$rchar_a (ptr), pointer)
;    1070  5
;    1071  4              end;
;    1072  4
;    1073  4          filenamedesc [dsc$w_length] = .pointer - filename;  ! Get final length
;    1074  3          end;
;    1075  3      !
;    1076  3      ! Now, copy filename into system pool space, and set up file control
;    1077  3      ! block descriptor to it.
;    1078  3      !
;    1079  3
;    1080  3          If Not Exe$Alononpaged (    !                                                    [10]
;    1081  3              Cblock [Ctl$W_File_Len] = .Filenamedesc [Dsc$W_Length]; !                    [10]
;    1082  3              CBlock [Ctl$W_File_Alloc],      !                                            [10]
;    1083  3              CBlock [Ctl$L_File_Ptr])        !                                            [10]
;    1084  3          Then                                !                                            [10]
;    1085  3              Return Rms$Error (Rms$_Dme);    ! Fatal If Can'T Allocate                    [10]
;    1086  3
;    1087  3          ch$move (.filenamedesc [dsc$w_length], .filenamedesc [dsc$a_pointer], .cblock [ctl$l_file_ptr]);
;    1088  3                                              ! Copy the string itself
;    1089  2          end;                                ! of file merge
;    1090  2
;    1091  2      !
;    1092  2      ! Now, access the file and set up static pointers, FAB fields, etc.
;    1093  2      !
;    1094  2          begin
;    1095  2
;    1096  2          local
;    1097  2              devlen,
;    1098  2              devptr : ref vector [, word],       ! Map to check fst 2 chars
;    1099  2              stat;
;    1100  2
;    1101  2          linkage
;    1102  2              jsb_device = jsb (register = 4, register = 5;   !                        [07]
;    1103  2                  register = 0, register  = 1);               !                        [07]
;    1104  2
;    1105  2          external routine
;    1106  2              scan$device : jsb_device novalue addressing_mode (long_relative);
;    1107  2
;    1108  2          scan$device (.filenamedesc [dsc$w_length],  !                                    [07]
;    1109  2                  .filenamedesc [dsc$a_pointer];      !                                    [07]
;    1110  2                  devlen, devptr);    ! Check for device name                          [07]
;    1111  2          cblock [ctl$w_unit] = .devlen<16, 16, 1>;   ! Store unit number returned
;    1112  2          stat =                      ! Do the device dependent part                   [07]
```

```
;    1112  4              begin                          !                                                                    [07]
;    1113  4
;    1114  4              local
;    1115  4                  AdapterType,               !                                                                    [10]
;    1116  4                  DeviceType,                !                                                                    [10]
;    1117  4                  DeviceUnit,                !                                                                    [10]
;    1118  4                  SlaveNumber,               !                                                                    [10]
;    1119  4                  Cinumber,                  !                                                                    [19]
;    1120  4                  OpenRoutine,               !                                                                    [10]
;    1121  4                  FileFormat,                ! Fileformat code                                                    [13]
;    1122  4                  InitByte,                  ! Address of byte to clear to cause driver init on file open         [13]
;    1123  4                  Status,                    ! Status return                                                      [13]
;    1124  4                  device : ref $ds_hpo_decl (),          ! Pointer to device PTABLE
;    1125  4                  controller : ref $ds_hpo_decl (),      ! Pointer to controller PTABLE
;    1126  4                  adapter : ref $ds_hpo_decl (),         ! Pointer to adapter PTABLE
;    1127  4                  PREV_CONTROLLER : REF $DS_HPO_DECL (),!Storage for previous controller ptable address       [22]
;    1128  4                  PREV_ADAPTER : REF $DS_HPO_DECL (); ! Storage for previous adapter ptable address          [22]
;    1129  4
;    1130  4              SlaveNumber = 0;               !                                                                    [10]
;    1131  4              OpenRoutine = 0;               !                                                                    [10]
;    1132  4              Cinumber    = 0;               !                                                                    [19]
;    1133  4              Loc$Ptable (.DevLen<0, 16>, .DevPtr, -1; Device);       ! Find the ptable                         [09]
;    1134  4
;    1135  4              If .Device Eql 0 Then Return Rms$Error (Rms$_Dev);      ! If none, error
;    1136  4
;    1137  4              Controller = Adapter = CBlock [Ctl$L_Ptable] = PREV_ADAPTER = PREV_CONTROLLER = .Device;! Output R1 = ptable
;    1138  4
;    1139  4            While .Adapter [Hp$A_Link] NeqA 0 Do                    ! Locate link Ptables
;    1140  5                Begin
;    1141  5                PREV_CONTROLLER = .CONTROLLER;                      ! Save copy of previous controller ptable address  [22]
;    1142  5                Controller = .Adapter;                              ! Shift links down
;    1143  5                PREV_ADAPTER =.ADAPTER;                             ! Save copy                                [22]
;    1144  5                Adapter = .Adapter [Hp$A_Link]
;    1145  4                End;
;    1146  5              BEGIN                                                                                              [22]
;    1147  5                BIND
;    1148  5                    DEV_TYPE = ADAPTER [HP$T_TYPE] : VECTOR [, BYTE];       ! Points to ptable _LINK field  [22]
;    1149  5
;    1150  5              ! The following IF statement checks if the ptable _TYPE field is an SBIA, if so the previously saved adapter  [22]
;    1151  5              ! address is loaded into ADAPTER. This is necessary for a xxx CPU since HUB is equal to ABUS , and a _LINK    [22]
;    1152  5              ! filed of zero would indicate that the present ptable is for an SBIA.                              [22]
;    1153  5
;    1154  5                    IF .DEV_TYPE [0] EQL 4                          ! Check for correct ASCII count of 4.      [22]
;    1155  5                    AND .(DEV_TYPE [1]) <0,32> EQL %ASCII 'SBIA'    ! Check for _TYPE filed of SBIA            [22]
;    1156  5                    THEN                                                                                       [22]
;    1157  6                      BEGIN                                                                                    [22]
;    1158  6                            CONTROLLER = .PREV_CONTROLLER;          ! Load 'previous' controller ptable address [22]
;    1159  6                            ADAPTER = .PREV_ADAPTER;                ! Load 'prevous' adapter ptable address    [22]
;    1160  6                      END                                                                                      [22]
;    1161  4              END;                                                                                               [22]
;    1162  4                  DeviceUnit = .Device [Hp$B_Drive];                  ! Copy device unit number          [10]
;    1163  4                  Status = DSX$Check_Support (.Device, AdapterType, DeviceType, FileFormat, InitByte);    !    [13]
;    1164  4
;    1165  4                  If Not .Status                                      ! If device not supported          [13]
;    1166  4                  Then                                                                                   [13]
;    1167  5                      Begin                                                                              [13]
;    1168  5                      Fab [Fab$L_Stv] = .Status;                      ! set secondary status             [13]
```

```
;    1169  5              Return RMS$Error (RMS$_Dev)                 ! Device error                                          [13]
;    1170  4              End;                                        !                                                       [13]
;    1171  4
;    1172  4          If .InitByte NeqA 0                             ! If init byte was specified in table                   [13]
;    1173  4          Then                                            !                                                       [13]
;    1174  4              (.InitByte)<0, 8> = 0;                      ! .. clear the byte                                     [13]
;    1175  4
;    1176  4          Case .FileFormat From 1 To 3 Of                 ! Case on file format                                   [13]
;    1177  4              Set                                         !                                                       [13]
;    1178  4              [RMS$K_Ansi] :                              ! .. ANSI file (magtape)                                [13]
;    1179  5                  Begin                                   !                                                       [13]
;    1180  5                  DeviceUnit = .Controller [Hp$B_Drive];  ! Modify device unit number                             [13]
;    1181  5                  SlaveNumber = .Device [Hp$B_Drive];     ! Set slave number                                      [13]
;    1182  5                  Fab [Fab$L_Dev] = Device$_Ansi;         ! Set device characteristics                            [13]
;    1183  5                  CBlock [Ctl$V_Ansi] = True;             ! Set ANSI bit                                          [13]
;    1184  5                  OpenRoutine = Ansi$Open                 ! Set OPEN routine address                              [13]
;    1185  4                  End;                                    !                                                       [13]
;    1186  4              [RMS$K_Ods] :                               ! .. ODS file (normal disk)                             [13]
;    1187  5                  Begin                                   !                                                       [13]
;    1188  5                  Cinumber = .Controller [Hp$B_Ci_tem] ;  ! Set CI port number for HSC50                          [20]
;    1189  5                  Fab [Fab$L_Dev] = Device$_Ods;          ! Set device characteristics                            [13]
;    1190  5                  Cblock [Ctl$V_Ods] = True;              ! Set ODS bit                                           [13]
;    1191  5                  OpenRoutine = Ods$Open                  ! Set OPEN routine address                              [13]
;    1192  4                  End;                                    !                                                       [13]
;    1193  4              [RMS$K_Rt11] :                              ! .. RT11 console media                                 [13]
;    1194  5                  Begin                                   !                                                       [13]
;    1195  5                  Fab [Fab$L_Dev] = Device$_Rt11;         ! .. set device characteristics                         [13]
;    1196  5                  CBlock [Ctl$V_Rt11] = True;             ! .. set RT-11 bit                                      [13]
;    1197  5                  OpenRoutine = Rt11$Open                 ! .. set OPEN routine address                           [13]
;    1198  4                  End;                                    !                                                       [13]
;    1199  4              [OutRange] :                                ! Bad support table file code                           [13]
;    1200  5                  Begin                                   !                                                       [13]
;    1201  5                  Fab [Fab$L_Stv] = DS$_ProgErr;          ! set secondary return status                           [13]
;    1202  5                  Return RMS$Error (RMS$_Dev)             ! error code                                            [13]
;    1203  5                  End                                     !                                                       [13]
;    1204  4              Tes;                                        !                                                       [13]
;    1205  4
;    1206  5          (If .OpenRoutine Neq 0                          ! Open file if good device                              [10]
;    1207  5          Then                                            !                                                       [10]
;    1208  6              Begin                                       !                                                       [10]
;    1209  6              CBlock [Ctl$L_RPB] = Dsr$Allocate_Pseudo_RPB (      !                                               [10]
;    1210  6                   DeviceType,                            !                                                       [10]
;    1211  6                  .AdapterType,                           !                                                       [10]
;    1212  6                  .Controller [HP$A_Device],              !                                                       [10]
;    1213  6                  .Adapter [HP$A_Device],                 !                                                       [10]
;    1214  6                  .DeviceUnit,                            !                                                       [10]
;    1215  6                  .SlaveNumber,                           !                                                       [10]
;    1216  6                  .Cinumber);                             !                                                       [10]
;    1217  6              (.OpenRoutine) ()                           ! Call the open routine                                 [10]
;    1218  6              End                                         !                                                       [10]
;    1219  5          Else                                            !                                                       [10]
;    1220  5              Rms$_Dev)                                   !                                                       [10]
;    1221  5
;    1222  3          End;                                            !                                                       [07]
;    1223  3
;    1224  3      rms$load_xab ();                                    ! Load the FHCXAB if any
;    1225  3      rms$error (.stat)                                   ! Return status in FAB & R0
```

```
;   1226  3          end                                   ! of file access block
;   1227  1          end;                                  ! Of RMS$OPEN


                                                           .EXTRN  SCAN$DEVICE

                                                           .ENTRY  RMS$OPEN, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-  ; 0943
                                                                   R11
                                                           MOVAB   -316(SP), SP
                                                           CLRQ    CBLOCK                                       ; 1003
                                                           MOVL    FABPTR, FAB                                  ; 1004
                                                           BSBB    VERIFY_FAB                                   ; 1006
                                                           BLBS    R0, 1$
                                                           MOVL    #99596, R0
                                                           RET
                                                   1$:     TSTB    29(FAB)                                      ; 1008
                                                           BEQL    2$
                                                           MOVL    #99852, R0
                                                           BRW     25$
                                                   2$:     CLRW    2(FAB)                                       ; 1010
                                                           MOVL    #1, I                                        ; 1011
                                                   3$:     TSTL    RMS$FILE_TABLE[I]                            ; 1012
                                                           BNEQ    4$
                                                           MOVL    I, FILE_INDEX
                                                           BRB     5$
                                                   4$:     AOBLEQ  #3, I, 3$
                                                           MNEGL   #1, FILE_INDEX                               ; 1011
                                                   5$:     CMPL    FILE_INDEX, #-1                              ; 1014
                                                           BEQL    6$
                                                           MOVZBL  #92, R1                                      ; 1018
                                                           JSB     EXE$ALONONPAGED
                                                           MOVL    R1, CBLOCKSIZE
                                                           MOVL    R2, CBLOCK
                                                           BLBS    R0, 7$
                                                   6$:     MOVL    #99540, 8(FAB)                               ; 1020
                                                           MOVL    #99540, R0
                                                           RET
                                                   7$:     MOVL    CBLOCK, RMS$FILE_TABLE[FILE_INDEX]           ; 1022
                                                           MOVC5   #0, (SP), #0, #92, (CBLOCK)                  ; 1023
                                                           MOVW    CBLOCKSIZE, 8(CBLOCK)                        ; 1024
                                                           MOVB    #1, (CBLOCK)                                 ; 1025
                                                           MOVW    FILE_INDEX, 2(FAB)                           ; 1026
                                                           CLRL    68(FAB)                                      ; 1032
                                                           CLRL    56(FAB)
                                                           CLRL    12(FAB)
                                                           CLRB    62(FAB)
                                                           CLRW    54(FAB)                                      ; 1031
                                                           MOVL    #65537, 8(FAB)                               ; 1033
                                                           MOVC5   #0, (SP), #0, #40, FILEDESC                  ; 1034
                                                           PUSHAB  FILEDESC                                     ; 1035
                                                           PUSHL   #17
                                                           CALLS   #3, BREAKUP_FILE
                                                           PUSHAB  FILEDESC                                     ; 1036
```

```
                                00000000G  EF   DD 000BC                PUSHL   DEF$Q_DEV+4                                    ; 1037
                             7E 00000000G  EF   3C 000C2                MOVZWL  DEF$Q_DEV, -(SP)                               ; 1036
                          0000G CF         03   FB 000C9                CALLS   #3, BREAKUP_FILE                               ;
                                      D8   AD   9F 000CE                PUSHAB  FILEDESC                                       ; 1038
                                00000000G  EF   DD 000D1                PUSHL   DEF$Q_DIR+4                                    ; 1039
                             7E 00000000G  EF   3C 000D7                MOVZWL  DEF$Q_DIR, -(SP)                               ; 1038
                          0000G CF         03   FB 000DE                CALLS   #3, BREAKUP_FILE                               ;
                                      D8   AD   9F 000E3                PUSHAB  FILEDESC                                       ; 1040
                                      30   AB   DD 000E6                PUSHL   48(FAB)                                        ; 1041
                             7E       35   AB   9A 000E9                MOVZBL  53(FAB), -(SP)                                 ; 1040
                          0000G CF         03   FB 000ED                CALLS   #3, BREAKUP_FILE                               ;
                                      D8   AD   9F 000F2                PUSHAB  FILEDESC                                       ; 1042
                                      2C   AB   DD 000F5                PUSHL   44(FAB)                                        ; 1043
                             7E       34   AB   9A 000F8                MOVZBL  52(FAB), -(SP)                                 ; 1042
                          0000G CF         03   FB 000FC                CALLS   #3, BREAKUP_FILE                               ;
                             52       18   AE   9E 00101                MOVAB   FILENAME, POINTER                              ; 1051
                             54       FC   8F   9A 00105                MOVZBL  #252, LENGTH                                   ; 1052
                          14 AE            52   D0 00109                MOVL    POINTER, FILENAMEDESC+4                        ; 1053
                                           51   D4 0010D                CLRL    I                                              ; 1055
                             50       D8 AD41   7E 0010F 8$:             MOVAQ   FILEDESC[I], R0                                ; 1059
                             53             60   3C 00114                MOVZWL  (R0), LEN                                      ; 1065
                             50       04   A0   D0 00117                MOVL    4(R0), PTR                                     ; 1066
                                           53   D7 0011B 9$:             DECL    LEN                                            ; 1068
                                           09   19 0011D                BLSS    10$                                            ;
                                           54   D7 0011F                DECL    LENGTH                                         ;
                                           05   19 00121                BLSS    10$                                            ;
                             82            80   90 00123                MOVB    (PTR)+, (POINTER)+                             ; 1069
                                           F3   11 00126                BRB     9$                                             ;
                          E3               51   04 F3 00128 10$:        AOBLEQ  #4, I, 8$                                      ;
                             50       18   AE   9E 0012C                MOVAB   FILENAME, R0                                   ; 1073
                          10 AE            52   A3 00130                SUBW3   R0, POINTER, FILENAMEDESC                      ;
                             51       10   AE   3C 00135                MOVZWL  FILENAMEDESC, R1                               ; 1081
                          48 A9            51   B0 00139                MOVW    R1, 72(CBLOCK)                                 ;
                                00000000G  EF   16 0013D                JSB     EXE$ALONONPAGED                                ; 1083
                          4A A9            51   B0 00143                MOVW    R1, 74(CBLOCK)                                 ; 1082
                          4C A9            52   D0 00147                MOVL    R2, 76(CBLOCK)                                 ; 1083
                             0A            50   E8 0014B                BLBS    R0, 11$                                        ;
                             50 000184D4   8F   D0 0014E                MOVL    #99540, R0                                     ; 1085
                                      012C 31   00155                   BRW     25$                                            ;
                          4C B9 14 BE      10   AE 28 00158 11$:        MOVC3   FILENAMEDESC, @FILENAMEDESC+4, @76(CBLOCK)     ; 1087
                             55       14   AE   D0 0015F                MOVL    FILENAMEDESC+4, R5                             ; 1107
                             54       10   AE   3C 00163                MOVZWL  FILENAMEDESC, R4                               ;
                                00000000G  EF   16 00167                JSB     SCAN$DEVICE                                    ;
                             53             50   D0 0016D                MOVL    R0, R3                                         ;
                          50               53   F0 8F 78 00170          ASHL    #-16, DEVLEN, R0                               ; 1110
                          02 A9            50   B0 00175                MOVW    R0, 2(CBLOCK)                                  ;
                                           56   D4 00179                CLRL    OPENROUTINE                                    ; 1131
                                           57   7C 0017B                CLRQ    CINUMBER                                       ; 1132
                             52            01   CE 0017D                MNEGL   #1, R2                                         ; 1133
                             50            53   3C 00180                MOVZWL  DEVLEN, R0                                     ;
                                00000000G  EF   16 00183                JSB     LOC$PTABLE                                     ;
                             54            51   D0 00189                MOVL    R1, R4                                         ;
                                           76   13 001EC                BEQL    18$                                            ; 1135
                             55            54   D0 0018E                MOVL    DEVICE, PREV_CONTROLLER                        ; 1137
                             51            54   D0 00191                MOVL    DEVICE, PREV_ADAPTER                           ;
                          3C A9            54   D0 00194                MOVL    DEVICE, 60(CBLOCK)                             ;
                             52            54   D0 00198                MOVL    DEVICE, ADAPTER                                ;
```

```
                                53          54  D0 0019B          MOVL    DEVICE, CONTROLLER                          ;
                             20 A2          D5 0019E 12$:        TSTL    32(ADAPTER)                                 ; 1139
                                0F          13 001A1             BEQL    13$                                         ;
                                55          53  D0 001A3         MOVL    CONTROLLER, PREV_CONTROLLER                 ; 1141
                                53          52  D0 001A6         MOVL    ADAPTER, CONTROLLER                         ; 1142
                                51          52  D0 001A9         MOVL    ADAPTER, PREV_ADAPTER                       ; 1143
                                52       20 A2  D0 001AC         MOVL    32(ADAPTER), ADAPTER                        ; 1144
                                           EC  11 001B0         BRB     12$                                         ;
                                50       26 A2  9E 001B2 13$:    MOVAB   38(ADAPTER), R0                             ; 1148
                                04          60  91 001B6         CMPB    (R0), #4                                    ; 1154
                                           10  12 001B9         BNEQ    14$                                         ;
                         41494253  8F       01  A0 D1 001BB     CMPL    1(R0), #1095320147                          ; 1155
                                           06  12 001C3         BNEQ    14$                                         ;
                                53          55  D0 001C5         MOVL    PREV_CONTROLLER, CONTROLLER                 ; 1158
                                52          51  D0 001C8         MOVL    PREV_ADAPTER, ADAPTER                       ; 1159
                                55       08 A4  9A 001CB 14$:    MOVZBL  11(DEVICE), DEVICEUNIT                      ; 1162
                                5E          DD  001CF            PUSHL   SP                                          ; 1163
                                08 AE       9F  001D1            PUSHAB  FILEFORMAT                                  ;
                                10 AE       9F  001D4            PUSHAB  DEVICETYPE                                  ;
                                18 AE       9F  001D7            PUSHAB  ADAPTERTYPE                                 ;
                                54          DD  001DA            PUSHL   DEVICE                                      ;
                          FABC  CF          05  FB  001DC        CALLS   #5, DSX$CHECK_SUPPORT                       ;
                                06          50  E8  001E1        BLBS    STATUS, 15$                                 ; 1165
                            0C  AB          50  D0  001E4        MOVL    STATUS, 12(FAB)                             ; 1168
                                           1A  11  001E8         BRB     18$                                         ; 1169
                                           6F  D5  001EA 15$:    TSTL    INITBYTE                                    ; 1172
                                           0   13  001EC         BEQL    16$                                         ;
                                        00 B    94  001EE        CLRB    @INITBYTE                                   ; 1174
                            02          01  04 AE  CF  001F1 16$:  CASEL   FILEFORMAT, #1, #2                        ; 1176
                          0049          0032       0017  001F6 17$:  .WORD  19$-17$,-                                 ;
                                                                           20$-17$,-                                   ;
                                                                           21$-17$                                     ;
                            0C  AB 00660020   8F  D0 001FC         MOVL    #6684704, 12(FAB)                           ; 1201
                                50 00018404   8F  D0 00204 18$:    MOVL    #99524, R0                                  ; 1202
                                           77  11 0020B         BRB     25$                                         ;
                                55       0B A3  9A 0020D 19$:    MOVZBL  11(CONTROLLER), DEVICEUNIT                  ; 1180
                                58       0B A4  9A 00211         MOVZBL  1?(DEVICE), SLAVENUMBER                     ; 1181
                             40 AB       4020   8F  3C 00215     MOVZWL  #16416, 64(FAB)                             ; 1182
                             01 A9              08  88 0021B     BISB2   #8, 1(CBLOCK)                               ; 1183
                                56 00000000G    EF  9E 0021F     MOVAB   ANSI$OPEN, OPENROUTINE                      ; 1184
                                           28  11 00226         BRB     22$                                         ;
                                57       34 A3  9A 00228 20$:    MOVZBL  52(CONTROLLER), CINUMBER                    ; 1188
                             40 AB 10004008    8F  D0 0022C     MOVL    #268451848, 64(FAB)                         ; 1189
                             01 A9              02  88 00234     BISB2   #2, 1(CBLOCK)                               ; 1190
                                56       0000G CF  9E 00238     MOVAB   DDS$OPEN, OPENROUTINE                       ; 1191
                                           11  11 0023D         BRB     22$                                         ;
                             40 AB 10004018    8F  D0 0023F 21$:    MOVL    #268451864, 64(FAB)                         ; 1195
                             01 A9              01  88 00247     BISB2   #1, 1(CBLOCK)                               ; 1196
                                56       0000G CF  9E 0024B     MOVAB   RT11$OPEN, OPENROUTINE                      ; 1197
                                           23  13 00250 22$:    BEQL    23$                                         ; 1206
                                57          DD  00252            PUSHL   CINUMBER                                    ; 1216
                                      0120  8F  BB  00254        PUSHR   #^M<R5,R8>                                  ; 1214
                                        18  A2  DD  00258        PUSHL   24(ADAPTER)                                 ; 1213
                                        18  A3  DD  0025B        PUSHL   24(CONTROLLER)                              ; 1212
                                        20  AE  DD  0025E        PUSHL   ADAPTERTYPE                                 ; 1211
                                        20  AE  DD  00261        PUSHL   DEVICETYPE                                  ; 1210
                          F9A8  CF              07  FB  00264    CALLS   #7, DSS$ALLOCATE_PSEUDO_RPB                 ;
```

```
                          38   A9                50   D0 00269          MOVL    #0, 56(CBLOCK)                          ;
                               66                00   FB 0026D          CALLS   #C, (OPENROUTINE)                       ; 1217
                               52                50   D0 00270          MOVL    R0, STAT                                ;
                                                 07   11 00273          BRB     24$                                     ;
                               52 000184C4       8F   D0 00275 23$:     MOVL    #99524, STAT                            ; 1206
                          FC4D CF                00   FB 0027C 24$:     CALLS   #0, RMS$LOAD_XAB                        ; 1224
                               50                52   D0 00281          MOVL    STAT, R0                                ; 1225
                                                F885  30 00284 25$:     BSBW    RMS$ERROR                               ;
                                                      04 00287          RET                                             ; 1227

; Routine Size:  648 bytes,    Routine Base:  CODE + 03EF

;   1278  1
```

```
:    1229  1     %sbttl 'RMS$CONNECT routine'
:    1230  1
:    1231  1     global routine rms$connect (rabptr) =
:    1232  1
:    1233  1     !++
:    1234  1     ! Functional description:
:    1235  1     !       Connect a RAB to a FAB.  This is relatively trivial; the validity
:    1236  1     !       of both RAB and FAB are verified, checks are made to be sure the
:    1237  1     !       file is open and no other RAB has been connected previously.
:    1238  1     !
:    1239  1     ! Inputs:
:    1240  1     !       rabptr          pointer to the RAB to connect
:    1241  1     !
:    1242  1     ! Implicit inputs:
:    1243  1     !       RAB [rab$l_fat] points to the FAB to connect this RAB to
:    1244  1     !       FAB [fab$w_ifi] indexes the rms$file_table to find the CBLOCK
:    1245  1     !
:    1246  1     ! Outputs:
:    1247  1     !       none
:    1248  1     !
:    1249  1     ! Implicit outputs:
:    1250  1     !       RAB [rab$l_sts] return status
:    1251  1     !       RAB [rab$l_stv] secondary status
:    1252  1     !
:    1253  1     ! Side Effects:
:    1254  1     !       some fields in the RAB and CBLOCK are modified
:    1255  1     !
:    1256  1     ! Return codes:
:    1257  1     !       RMS$_NORMAL     all OK
:    1258  1     !       RMS$_CCR        a RAB is already connected to this FAB
:    1259  1     !       RMS$_RAB        the RAB block is invalid
:    1260  1     !       RMS$_FAB        the FAB block is invalid
:    1261  1     !       RMS$_IFI        the FAB's IFI field is invalid.
:    1262  1     !       RMS$_RAC        Invalid record access mode--only SEQuential and
:    1263  1     !                       RFA access modes are supported.
:    1264  1     !--
:    1265  1
:    1266  2          begin
:    1267  2
:    1268  2          global register
:    1269  2              rab = 10 : ref bblock ,                 ! Pointer to RAB
:    1270  2              fab = 11 : ref bblock ,                 ! Pointer to FAB
:    1271  2              cblock = 9 : ref bblock;                ! Pointer to CBLOCK
:    1272  2
:    1273  2          cblock = 0;
:    1274  2          fab = 0;
:    1275  2          rab = .rabptr;                              ! Map to RAB
:    1276  2
:    1277  2          if not verify_rab ()                        ! If the RAB is bad
:    1278  2          then
:    1279  2              return (rms$_rab);                      !  return error
:    1280  2
:    1281  2          rab [rab$l_stv] = 0;
:    1282  2
:    1283  2          if .rab [rab$b_rac] neq rab$c_seq           ! If not sequential access
:    1284  2              and .rab [rab$b_rac] neq rab$c_rfa      !  and not RFA random
:    1285  2          then
```

```
;    1286  2          return rab$error (rms$_rac);          ! then bad access mode
;    1287  2
;    1288  2      fab = .rab [rab$l_fab];                   ! Map to FAB
;    1289  2
;    1290  2      if not verify_fab ()                      ! If the FAB is bad
;    1291  2      then
;    1292  2          return rab$error (rms$_fab);          !  return error
;    1293  2
;    1294  2      if not verify_cblock ()                   ! If the IFI is bad
;    1295  2      then
;    1296  2          return rab$error (rms$_ifi);          !  return with error
;    1297  2
;    1298  2      if .cblock [ctl$l_rab] neq 0              ! If there's already a RAB
;    1299  2      then
;    1300  2          return rab$error (rms$_ccr);          !  it's no good--only one allowed
;    1301  2
;    1302  2      cblock [ctl$l_rab] = .rab;                ! Now set point to RAB
;    1303  2      cblock [ctl$l_vbn] = 1;                   ! Set internal RFA
;    1304  2      cblock [ctl$w_byte] = 0;
;    1305  2      cblock [ctl$w_rec_size] = 0;              ! Clear last-record size
;    1306  2      ch$fill (0, 6, rab [rab$w_rfa]);          ! Set RFA to 0
;    1307  2      rab [rab$l_rbf] = 0;
;    1308  2      rab [rab$w_rsz] = 0;
;    1309  2      cblock [ctl$l_nbp] = 0;
;    1310  2      rab$error (rms$_normal)                   ! Set good return
;    1311  1      end;

                     0FFC 00000          .ENTRY  RMS$CONNECT, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1231
                                                 R10,R11
                   59 D4 00002           CLRL    CBLOCK                                      ; 1273
                   58 D4 00004           CLRL    FAB                                         ; 1274
5A       04        AC D0 00006           MOVL    RABPTR, RAB                                 ; 1275
             FD58  30 0000A              BSBW    VERIFY_RAB                                  ; 1277
08                 50 E8 0000D           BLBS    R0, 1$
50 0001863C        8F D0 00010           MOVL    #99900, R0                                  ; 1279
                   04 00017              RET
          0C       AA D4 00018 1$:       CLRL    12(RAB)                                     ; 1281
          1E       AA 95 0001B           TSTB    30(RAB)                                     ; 1283
                   0F 13 0001E           BEQL    2$
02        1E       AA 91 00020           CMPB    30(RAB), #2                                 ; 1284
                   09 13 00024           BEQL    2$
50 00018644        8F D0 00026           MOVL    #99908, R0                                  ; 1286
                   4F 11 00029           BRB     6$
5B        3C       AA D0 0002F 2$:       MOVL    60(RAB), FAB                                ; 1288
             FCFE  30 00033              BSBW    VERIFY_FAB                                  ; 1290
09                 50 E8 00036           BLBS    R0, 3$
50 0001850C        8F D0 00039           MOVL    #99596, R0                                  ; 1292
                   3C 11 00040           BRB     6$
             FD02  30 00042 3$:          BSBW    VERIFY_CBLOCK                               ; 1294
09                 50 E8 00045           BLBS    R0, 4$
50 00018564        8F D0 00048           MOVL    #99684, R0                                  ; 1296
                   2D 11 0004F           BRB     6$
          20       A9 D5 00051 4$:       TSTL    32(CBLOCK)                                  ; 1298
```

```
                                        09  13 00054          BEQL    5$                                          ;
                                50 00018494  8F  D0 00056          MOVL    #99476, R0                                  ; 1300
                                             1F  11 0005D          BRB     6$                                          ;
                             20  A9          5A  D0 0005F 5$:      MOVL    RAB, 32(CBLOCK)                             ; 1302
                             18  A9          01  7D 00063          MOVQ    #1, 24(CBLOCK)                              ; 1303
          06          00         6E          00  2C 00067          MOVC5   #0, (SP), #0, #6, 16(RAB)                   ; 1306
                                         10  AA      0006C                                                           ;
                                         28  AA  D4  0006E          CLRL    40(RAB)                                     ; 1307
                                         22  AA  B4  00071          CLRW    34(RAB)                                     ; 1308
                                         04  A9  D4  00074          CLRL    4(CBLOCK)                                   ; 1309
                                50 00010001  8F  D0 00077          MOVL    #65537, R0                                  ; 1310
                                         FCA9  30 0007E 6$:        BSBW    RAB$ERROR                                   ;
                                               04 00081          RET                                                 ; 1311

; Routine Size:  130 bytes,    Routine Base:  CODE + 0677

;   1312  1
```

```
;    1313  1      %sbttl 'RMS routine'
;    1314  1
;    1315  1      global routine rms$get (rabptr) =
;    1316  1
;    1317  1      !++
;    1318  1      ! Functional description:
;    1319  1      !       Read the next record into a user-specified buffer, from the
;    1320  1      !       file currently open on the FAB to which the input RAB is
;    1321  1      !       connected.
;    1322  1      !
;    1323  1      ! Inputs:
;    1324  1      !       rabptr                  pointer to the RAB to use
;    1325  1      !
;    1326  1      ! Implicit inputs:
;    1327  1      !       RAB [rab$l_fat]         The FAB which controls the file
;    1328  1      !       RAB [rab$w_rfa]         Desired record's file address
;    1329  1      !       FAB [fab$w_ifi]         Index in RMS$FILE_TABLE for this file
;    1330  1      !       CBLOCK                  Internal control block
;    1331  1      !
;    1332  1      ! Outputs:
;    1333  1      !       none
;    1334  1      !
;    1335  1      ! Implicit outputs:
;    1336  1      !       The RBF and RSZ fields of the RAB are set to describe the
;    1337  1      !       record read.
;    1338  1      !
;    1339  1      !       RAB [rab$l_sts]         contains the status code
;    1340  1      !       RAB [rab$l_stv]         may contain a secondary status
;    1341  1      !       RAB [rab$w_rfa]         in sequential access mode ONLY, this
;    1342  1      !                               3-word field is updated prior to the actual
;    1343  1      !                               access.
;    1344  1      !
;    1345  1      ! Side effects:
;    1346  1      !       CBLOCK                  The next RFA, the cache control data, and
;    1347  1      !                               the tape position control are all subject
;    1348  1      !                               to change.
;    1349  1      !
;    1350  1      ! Return codes:
;    1351  1      !       RMS$_NORMAL             Everything's nice
;    1352  1      !       RMS$_RER                Some read error (RAB [rab$l_stv] contains
;    1353  1      !                               the actual SS$_* error code)
;    1354  1      !       RMS$_RFA                Invalid RFA was specified in random-by-RFA
;    1355  1      !                               mode.
;    1356  1      !       RMS$_EOF                Attempt to read record past end of file
;    1357  1      !       RMS$_RAB                bad RAB
;    1358  1      !       RMS$_FAB                bad FAB
;    1359  1      !       RMS$_IFI                bad IFI (or CBLOCK)
;    1360  1      !       RMS$_ISI                FAB and RAB not connected
;    1361  1      !       RMS$_RTB                Record too big for buffer (truncated)
;    1362  1      !--
;    1363  1
;    1364  2          begin
;    1365  2
;    1366  2          global register
;    1367  2              cblock = 9 : ref bblock ,           ! Map to CBLOCK
;    1368  2              rab = 10 : ref bblock ,             ! Map to RAB
;    1369  2              fab = 11 : ref bblock;              ! Map to FAB
```

```
;    1370  2
;    1371  2          local
;    1372  2              stat;
;    1373  2
;    1374  2          fab = cblock = 0;
;    1375  2          rab = .rabptr;                          ! Get pointer to RAB
;    1376  2
;    1377  2          if not verify_rab ()                    ! If it isn't a RAB
;    1378  2          then
;    1379  2              return (rms$_rab);                  !  no good at all
;    1380  2
;    1381  2          rab [rab$l_stv] = 0;                    ! Clear out STV field
;    1382  2          fab = .rab [rab$l_fab];                 ! Point to RAB's FAB
;    1383  2
;    1384  2          if not verify_fab ()                    ! If it isn't a FAB
;    1385  2          then
;    1386  2              return rab$error (rms$_fab);        !  then quit nicely
;    1387  2
;    1388  2          if not verify_cblock ()                 ! Load CBLOCK pointer,
;    1389  2          then
;    1390  2              return rab$error (rms$_ifi);        !  and quit if IFI's bad
;    1391  2
;    1392  2          if .cblock [ctl$l_rab] neq .rab         ! If this isn't the right RAB
;    1393  2          then
;    1394  2              return rab$error (rms$_isi);        !  say 'bad stream id'
;    1395  2
;    1396  2      ;
;    1397  2      ! Now, all the control stuff is OK.  We can go to work
;    1398  2      !
;    1399  2          cblock [ctl$b_op] = ctl$c_get;          ! We're on GET operation
;    1400  2
;    1401  2          if .rab [rab$b_rac] eql rab$c_rfa       ! If random, copy user's
;    1402  2          then
;    1403  2              ch$move (6, rab [rab$w_rfa], cblock [ctl$w_rfa]);       ! RFA
;    1404  2
;    1405  2          rab [rab$w_rsz] = rab [rab$l_rbf] = 0;      ! Init context                          [05]
;    1406  2
;    1407  2          if .cblock [ctl$l_vbn] gtr .cblock [ctl$l_eofvbn]   ! If RFA is past EOF
;    1408  2              or .cblock [ctl$l_vbn] eql .cblock [ctl$l_eofvbn]       !
;    1409  2              and .cblock [ctl$w_byte] geq .cblock [ctl$w_offbyte]
;    1410  2          then
;    1411  2              return rab$error (
;    1412  2
;    1413  2                      if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof); !  don't bother
;    1414  2
;    1415  2          stat = (.cblock [ctl$l_get]) ();        ! Call actual GET routine
;    1416  2          ch$move (6, cblock [ctl$w_rfa], rab [rab$w_rfa]);   ! Set user RFA
;    1417  2          rab$error (.stat)                       ! Return whatever status
;    1418  1          end;                                    ! Of RMS$GET


                  OFFC 00000          .ENTRY  RMS$GET, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- ; 1315
                                              R11                                         ;
                59  D4 00002          CLRL    CBLOCK                                      ; 1374
```

```
                                    5B  D4 00004         CLRL    FAB                                    ;
                      5A        04  AC  D0 00006         MOVL    RABPTR, RAB                            ; 1375
                                  FCD6  30 0000A         BSBW    VERIFY_RAB                             ; 1377
                      08            50  E8 0000D         BLBS    R0, 1$                                 ;
                      50 0001863C   8F  D0 00010         MOVL    #99900, R0                             ; 1379
                                        04 00017         RET                                            ;
                                0C  AA  D4 00018 1$:     CLRL    12(RAB)                                ; 1381
                      5B        3C  AA  D0 0001B         MOVL    60(RAB), FAB                           ; 1382
                                  FC90  30 0001F         BSBW    VERIFY_FAB                             ; 1384
                      09            50  E8 00022         BLBS    R0, 2$                                 ;
                      50 0001850C   8F  D0 00025         MOVL    #99596, R0                             ; 1386
                                    6B  11 0002C         BRB     9$                                     ;
                                  FC94  30 0002E 2$:     BSBW    VERIFY_CBLOCK                          ; 1388
                      09            50  E8 00031         BLBS    R0, 3$                                 ;
                      50 00018564   8F  D0 00034         MOVL    #99684, R0                             ; 1390
                                    5C  11 0003B         BRB     9$                                     ;
                      5A        20  A9  D1 0003D 3$:     CMPL    32(CBLOCK), RAB                        ; 1392
                                    09  13 00041         BEQL    4$                                     ;
                      50 00018584   8F  D0 00043         MOVL    #99716, R0                             ; 1394
                                    4D  11 0004A         BRB     9$                                     ;
                      69            02  90 0004C 4$:     MOVB    #2, (CBLOCK)                           ; 1399
                      02        1E  AA  91 0004F         CMPB    30(RAB), #2                            ; 1401
                                    06  12 00053         BNEQ    5$                                     ;
        18    A9      10  AA        06  28 00055         MOVC3   #6, 16(RAB), 24(CBLOCK)                ; 1403
                                28  AA  D4 0005B 5$:     CLRL    40(RAB)                                ; 1405
                                22  AA  B4 0005E         CLRW    34(RAB)                                ;
                    44    A9    18  A9  D1 00061         CMPL    24(CBLOCK), 68(CBLOCK)                 ; 1407
                                    09  14 00066         BGTR    6$                                     ;
                                    1F  12 00068         BNEQ    8$                                     ; 1408
                    42    A9    1C  A9  B1 0006A         CMPW    28(CBLOCK), 66(CBLOCK)                 ; 1409
                                    18  1F 0006F         BLSSU   8$                                     ;
                      02        1E  AA  91 00071 6$:     CMPB    30(RAB), #2                            ; 1413
                                    09  12 00075         BNEQ    7$                                     ;
                      50 0001865C   8F  D0 00077         MOVL    #99932, R0                             ;
                                    19  11 0007E         BRB     9$                                     ;
                      50 0001827A   8F  D0 00080 7$:     MOVL    #98938, R0                             ;
                                    10  11 00087         BRB     9$                                     ;
                    0C    B9        00  FB 00089 8$:     CALLS   #0, @12(CBLOCK)                        ; 1415
                          56        50  D0 0008D         MOVL    R0, STAT                               ;
        10    AA      18  A9        06  28 0009C         MOVC3   #6, 24(CBLOCK), 16(RAB)                ; 1416
                          50        56  D0 00096         MOVL    STAT, R0                               ; 1417
                                  FC0C  30 00099 9$:     BSBW    RAB$ERROR                              ;
                                        04 0009C         RET                                            ; 1418

; Routine Size:  157 bytes,    Routine Base:  CODE + 06F9


;    1419  1
```

```
;    1420  1      %sbttl 'RMS$READ routine'
;    1421  1
;    1422  1      global routine rms$read (rabptr) =
;    1423  1
;    1424  1      !++
;    1425  1      ! Functional description:
;    1426  1      !       Read from a file, beginning with either the next or some specified
;    1427  1      !       VBN.  As many blocks are read as will fit in the user buffer
;    1428  1      !
;    1429  1      ! Inputs:
;    1430  1      !       rabptr                  pointer to the RAB to use
;    1431  1      !
;    1432  1      ! Implicit inputs:
;    1433  1      !       RAB [rab$l_fab]         The FAB which controls the file
;    1434  1      !       RAB [rab$l_bkt]         VBN to read from
;    1435  1      !       FAB [fab$w_ifi]         Index in RMS$FILE_TABLE for this file
;    1436  1      !       CBLOCK                  Internal control block
;    1437  1      !
;    1438  1      ! Outputs:
;    1439  1      !       none
;    1440  1      !
;    1441  1      ! Implicit outputs:
;    1442  1      !       The RBF and RSZ fields of the RAB are set to describe the
;    1443  1      !       record read.
;    1444  1      !
;    1445  1      !       RAB [rab$l_sts]         contains the status code
;    1446  1      !       RAB [rab$l_stv]         may contain a secondary status
;    1447  1      !
;    1448  1      ! Side effects:
;    1449  1      !       CBLOCK                  The next block, the cache control data, and
;    1450  1      !                               the tape position control are all subject
;    1451  1      !                               to change.
;    1452  1      !
;    1453  1      ! Return codes:
;    1454  1      !       RMS$_NORMAL             Everything's nice
;    1455  1      !       RMS$_RER                Some read error (RAB [rab$l_stv] contains
;    1456  1      !                               the actual SS$_* error code)
;    1457  1      !       RMS$_RAB                bad RAB
;    1458  1      !       RMS$_FAB                bad FAB
;    1459  1      !       RMS$_IFI                bad IFI (or CBLOCK)
;    1460  1      !       RMS$_ISI                FAB and RAB not connected
;    1461  1      !       RMS$_EOF                VBN specified is past EOF
;    1462  1      !--
;    1463  1
;    1464  2          begin
;    1465  2
;    1466  2          global register
;    1467  2              cblock = 9 : ref bblock ,           ! Map to CBLOCK
;    1468  2              rab = 10 : ref bblock ,             ! Map to RAB
;    1469  2              fab = 11 : ref bblock;              ! Map to FAB
;    1470  2
;    1471  2          local
;    1472  2              stat;
;    1473  2
;    1474  2          fab = cblock = 0;
;    1475  2          rab = .rabptr;                          ! Get pointer to RAB
;    1476  2
```

```
;    1477  2          if not verify_rab ()                    ! If it isn't a RAB
;    1478  2          then
;    1479  2              return (rms$_rab);                  !  no good at all
;    1480  2
;    1481  2          rab [rab$l_stv] = 0;                    ! Clear out STV field
;    1482  2          fab = .rab [rab$l_fab];                 ! Point to RAB's FAB
;    1483  2
;    1484  2          if not verify_fab ()                    ! If it isn't a FAB
;    1485  2          then
;    1486  2              return rab$error (rms$_fab);        !  then quit nicely
;    1487  2
;    1488  2          if not verify_cblock ()                 ! Load CBLOCK pointer,
;    1489  2          then
;    1490  2              return rab$error (rms$_ifi);        !  and quit if IFI's bad
;    1491  2
;    1492  2          if .cblock [ctl$l_rab] neq .rab         ! If this isn't the right RAB
;    1493  2          then
;    1494  2              return rab$error (rms$_isi);        !  say 'bad stream id'
;    1495  2
;    1496  2      !
;    1497  2      ! Now, all the control stuff is OK.  We can go to work
;    1498  2      !
;    1499  2          cblock [ctl$b_op] = ctl$c_read;         ! We're on READ operation
;    1500  2
;    1501  2          if .cblock [ctl$l_nbp] eql 0            ! If this is first
;    1502  2          then
;    1503  2              cblock [ctl$l_nbp] = 1;             ! Set to block 1
;    1504  2
;    1505  2          if .rab [rab$l_bkt] gtr 0               ! If user specified VBN
;    1506  2          then
;    1507  2              cblock [ctl$l_nbp] = .rab [rab$l_bkt];  ! Copy to next block pointer
;    1508  2
;    1509  2          rab [rab$l_rfa0] = .cblock [ctl$l_nbp];     ! Set RFA to beginning of block
;    1510  2          rab [rab$w_rfa4] = 0;
;    1511  2          rab [rab$w_rsz] = rab [rab$l_rbf] = 0;      ! Init context stuff                    [05]
;    1512  2
;    1513  2          if .cblock [ctl$l_nbp] gtr .cblock [ctl$l_eofvbn]   ! If past end of file
;    1514  3              or (.cblock [ctl$l_nbp] eql .cblock [ctl$l_eofvbn]      !                      [05]
;    1515  3               and .cblock [ctl$w_offbyte] eql 0)     !  (check if on null block)            [05]
;    1516  2          then
;    1517  2              return rab$error (rms$_eof);        !  don't bother
;    1518  2
;    1519  2          stat = (.cblock [ctl$l_read]) ();       ! Call actual READ routine
;    1520  2          rab$error (.stat)                       ! Return whatever status
;    1521  1          end;                                    ! Of RMS$READ


                                 0FFC 00000          .ENTRY  RMS$READ, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1422
                                                             R11                                          ;
                               59  D4 00002          CLRL    CBLOCK                                       ; 1474
                               5B  D4 00004          CLRL    FAB                                          ;
                     5A    04  AC  D0 00006          MOVL    RABPTR, RAB                                  ; 1475
                         FC39  30 0000A              BSBW    VERIFY_RAB                                   ; 1477
                     08        50  E8 0000D          BLBS    R0, 1$                                       ;
```

```
                50 0001863C   8F   D0 00010          MOVL    #99900, R0                      ; 1479
                              04   00017          RET                                     ;
                      0C      AA   D4 00018 1$:     CLRL    12(RAB)                         ; 1481
                5B    3C      AA   D0 0001B         MOVL    60(RAB), FAB                    ; 1482
                            FBF3   30 0001F         BSBW    VERIFY_FAB                      ; 1484
                09              50   E8 00022      BLBS    R0, 2$                          ;
                50 0001850C   8F   D0 00025         MOVL    #99596, R0                      ; 1486
                              5D   11 0002C         BRB     9$                              ;
                            FBF7   30 0002E 2$:     BSBW    VERIFY_CBLOCK                   ; 1488
                09              50   E8 00031      BLBS    R0, 3$                          ;
                50 00018564   8F   D0 00034         MOVL    #99684, R0                      ; 1490
                              4E   11 0003B         BRB     9$                              ;
                5A      20    A9   D1 0003D 3$:     CMPL    32(CBLOCK), RAB                 ; 1492
                              09   13 00041         BEQL    4$                              ;
                50 00018584   8F   D0 00043         MOVL    #99716, R0                      ; 1494
                              3F   11 0004A         BRB     9$                              ;
                69            03   90 0004C 4$:     MOVB    #3, (CBLOCK)                    ; 1499
                        04    A9   D5 0004F         TSTL    4(CBLOCK)                       ; 1501
                              04   12 00052         BNEQ    5$                              ;
          04    A9            01   D0 00054         MOVL    #1, 4(CBLOCK)                   ; 1503
                      38      AA   D5 00058 5$:     TSTL    56(RAB)                         ; 1505
                              05   15 0005B         BLEQ    6$                              ;
          04    A9      38    AA   D0 0005D         MOVL    56(RAB), 4(CBLOCK)              ; 1507
          10    AA      04    A9   D0 00062 6$:     MOVL    4(CBLOCK), 16(RAB)              ; 1509
                        14    AA   B4 00067         CLRW    20(RAB)                         ; 1510
                        28    AA   D4 0006A         CLRL    40(RAB)                         ; 1511
                        22    AA   B4 0006D         CLRW    34(RAB)                         ;
          44    A9      04    A9   D1 00070         CMPL    4(CBLOCK), 68(CBLOCK)           ; 1513
                              07   14 00075         BGTR    7$                              ;
                              0E   12 00077         BNEQ    8$                              ; 1514
                        42    A9   B5 00079         TSTW    66(CBLOCK)                      ; 1515
                              09   12 0007C         BNEQ    8$                              ;
                50 0001827A   8F   D0 0007E 7$:     MOVL    #98938, R0                      ; 1517
                              04   11 00085         BRB     9$                              ;
          10    B9            00   FB 00087 8$:     CALLS   #0, @16(CBLOCK)                 ; 1519
                            FB7D   30 0008B 9$:     BSBW    RAB$ERROR                       ; 1520
                                   04 0008E         RET                                     ; 1521

; Routine Size:  143 bytes,    Routine Base:  CODE + 0796

;   1522  1
```

```
;    1523  1      %sbttl 'RMS$DISCONNECT routine'
;    1524  1
;    1525  1      global routine rms$disconnect (rabptr) =
;    1526  1
;    1527  1      !++
;    1528  1      ! Functional description:
;    1529  1      !       Disconnect a RAB from a FAB.  This is relatively trivial; the validity
;    1530  1      !       of both RAB and FAB are verified, checks are made to be sure the
;    1531  1      !       file is open and the RAB was connected to that FAB
;    1532  1      !
;    1533  1      ! Inputs:
;    1534  1      !       rabptr          pointer to the RAB to disconnect
;    1535  1      !
;    1536  1      ! Implicit inputs:
;    1537  1      !       RAB [rab$l_fab] points to the FAB to disconnect this RAB from
;    1538  1      !       FAB [fab$w_ifi] indexes the rms$file_table to find the CBLOCK
;    1539  1      !
;    1540  1      ! Outputs:
;    1541  1      !       none
;    1542  1      !
;    1543  1      ! Implicit outputs:
;    1544  1      !       RAB [rab$l_sts] return status
;    1545  1      !       RAB [rab$l_stv] secondary status
;    1546  1      !
;    1547  1      ! Side Effects:
;    1548  1      !       some fields in the RAB and CBLOCK are modified
;    1549  1      !
;    1550  1      ! Return codes:
;    1551  1      !       RMS$_NORMAL     all OK
;    1552  1      !       RMS$_RAB        the RAB block is invalid
;    1553  1      !       RMS$_FAB        the FAB block is invalid
;    1554  1      !       RMS$_IFI        the FAB's IFI field is invalid.
;    1555  1      !       RMS$_ISI        RAB and FAB weren't connected
;    1556  1      !--
;    1557  1
;    1558  2          begin
;    1559  2
;    1560  2          global register
;    1561  2              rab = 10 : ref bblock ,                 ! Pointer to RAB
;    1562  2              fab = 11 : ref bblock ,                 ! Pointer to FAB
;    1563  2              cblock = 9 : ref bblock;                ! Pointer to CBLOCK
;    1564  2
;    1565  2          fab = cblock = 0;
;    1566  2          rab = .rabptr;                              ! Map to RAB
;    1567  2
;    1568  2          if not verify_rab ()                        ! If the RAB is bad
;    1569  2          then
;    1570  2              return (rms$_rab);                      !  return error
;    1571  2
;    1572  2          rab [rab$l_stv] = 0;
;    1573  2          fab = .rab [rab$l_fab];                     ! Map to FAB
;    1574  2
;    1575  2          if not verify_fab ()                        ! If the FAB is bad
;    1576  2          then
;    1577  2              return rab$error (rms$_fab);            !  return error
;    1578  2
;    1579  2          if not verify_cblock ()                     ! If bad IFI
```

B 1 QA$Find_User_Call_Frame
C 1 QA$Find_User_Call_Frame
D 1 QA$Init
E 1 QA$Init
F 1 QA$Main
G 1 QA$Main
H 1 QA$Main
I 1 QA$Main
J 1 QA$Main
K 1 QA$Main
L 1 QA$Main
M 1 QA$Main
N 1 QA$Main
B 2 QA$Main
C 2 QA$Main
D 2 QA$Print_Forced_Errors
E 2 Dispose
F 2 Dispose
G 2 Dispose
H 2 Dispose
I 2 Dispose
J 2 Dispose
K 2 QA$Print_Overall_Errors
L 2 QA$Print_Overall_Errors
M 2 QA$Print_Overall_Errors
N 2 QA$Print_Overall_Errors
B 3 QA$Print_Overall_Errors
C 3 QA$Print_Overall_Errors
D 3 QA$Print_Overall_Errors
E 3 End of Module QAMAIN
F 3
G 3 QIO
H 3 07-18
I 3
J 3 Libraries and Equated Symbols
K 3 Libraries and Equated Symbols
L 3 External Symbols
M 3 External Symbols
N 3 External Symbols
B 4 External Symbols
C 4 EXE$QIO QIO DISPATCHING ROUTI
D 4 EXE$QIO QIO DISPATCHING ROUTI
E 4 QIO$INITIALIZE Initialize QIO
F 4 QIO$INITIALIZE Initialize QIO
G 4 QIO$CLEANUP Remove old QIO dat
H 4 QIO$CLEANUP Remove old QIO dat
I 4 QIO$CLEANUP Remove old QIO dat
J 4 QIO$ADDUNIT Add a unit to the
K 4 QIO$ADDUNIT Add a unit to the
L 4 QIO$ADDUNIT Add a unit to the
M 4 QIO$ADDUNIT Add a unit to the
N 4 QIO$ADDUNIT Add a unit to the
B 5 QIO$ADDUNIT Add a unit to the
C 5 QIO$ADDUNIT Add a unit to the
D 5 INI_MBADP BUILD ADP AND INITIA
E 5 INI_MBADP BUILD ADP AND INITIA
F 5 INI_UBADP BUILD ADP AND INITIA
G 5 INI_DRADP BUILD ADP AND INITIA
H 5 INI_DRADP BUILD ADP AND INITIA
I 5 MAXIMIZE ACCESS MODE
J 5 BUG$CHECK Bugcheck notificatio
K 5 QIO$LOAD_DB Insert unit into d
L 5 QIO$LOAD_DB Insert unit into d
M 5 QIO$LOAD_DB Insert unit into d
N 5 QIO$LOAD_DB Insert unit into d
B 6 QIO$LOAD_DB Insert unit into d
C 6 QIO$LOAD_DB Insert unit into d
D 6 DB_ERROR ERROR LOADING DATABAS
E 6 DB_ERROR ERROR LOADING DATABAS
F 6 IOGEN$CNTRL_INI Initialize a c
G 6 MEMORY ALLOCATION SUBROUTINES
H 6 ADPLINK Link adapter to end o
I 6 Exe$PwrTimChk - Check reasonab
J 6 Exe$PwrTimChk - Check reasonab
K 6 Symbol table
L 6 Symbol table
M 6 Symbol table
N 6 Symbol table
B 7 Symbol table
C 7 Symbol table
D 7 Psect synopsis
E 7 Cross reference
F 7 Cross reference
G 7 Cross reference
H 7 Cross reference
I 7 Cross reference
J 7 Cross reference
K 7 Cross reference
L 7 Cross reference
M 7 Cross reference
N 7 Cross reference
B 8 *** QIOFDT function dispatch r
C 8 *** QIOFDT function dispatch r
D 8 *** QIOFDT function dispatch r
E 8 ONE PARAMETER FUNCTION PROCESS
F 8 ZERO PARAMETER FUNCTION PROCES
G 8 READ AND WRITE FUNCTION PROCES
H 8 READ AND WRITE FUNCTION BUFFER
I 8 READ AND WRITE BUFFER CHECK AN
J 8 READ AND WRITE BUFFER CHECK AN
K 8 CHECK BUFFER ACCESSIBILITY FOR
L 8 CHECK BUFFER ACCESSIBILITY FOR
M 8 CHECK BUFFER ACCESSIBILITY FOR
N 8 CHECK BUFFER ACCESSIBILITY FOR
B 9 SET DEVICE MODE AND CHARACTERI
C 9 SET DEVICE MODE AND CHARACTERI
D 9 SENSE DEVICE MODE AND CHARACTE
E 9 CARRIAGE CONTROL INTERPRETATIO
F 9 CARRIAGE CONTROL INTERPRETATIO
G 9 Symbol table
H 9 Symbol table
I 9 Symbol table
J 9 Cross reference
K 9 Cross reference
L 9 Cross reference
M 9 Cross reference
N 9 *** QIOREQ handle QIO request
B 10 *** QIOREQ handle QIO request
C 10 *** QIOREQ handle QIO request
D 10 DECLARATIONS
E 10 QUEUE I/O REQUEST
F 10 QUEUE I/O REQUEST
G 10 QUEUE I/O REQUEST
H 10 QUEUE I/O REQUEST
I 10 QUEUE I/O REQUEST
J 10 COMPLETE I/O OPERATION
K 10 QUEUE I/O PACKET TO DRIVER
L 10 EXE$ALTQUEPKT - Call driver AL
M 10 QUEUE I/O PACKET TO ACP
N 10 INSERT I/O PACKET IN UNIT QUEU
B 11 INSERT I/O PACKET IN QUEUE BY
C 11 Symbol table
D 11 Symbol table
E 11 Symbol table
F 11 Symbol table
G 11 Psect synopsis
H 11 Cross reference
I 11 Cross reference
J 11 Cross reference
K 11 Cross reference
L 11 *** RELOCDRV relocate I/O driv
M 11 *** RELOCDRV relocate I/O driv
N 11 INITIALIZE DRIVER DATA BASE
B 12 REINITIALIZE DRIVER DATA BASE
C 12 LOCAL SUBROUTINE TO EXECUTE OP
D 12 LOCAL SUBROUTINE TO EXECUTE OP
E 12 LOCAL SUBROUTINE TO LOCATE DAT
F 12 LOCAL SUBROUTINE TO LOCATE DAT
G 12 Symbol table
H 12 Symbol table
I 12 Cross reference
J 12 Cross reference
K 12 *** RMS Master RMS routines
L 12 *** RMS Master RMS routines
M 12 *** RMS Master RMS routines
N 12 *** RMS Master RMS routine
B 13 Device support table
C 13 Allocate pseudo RPB
D 13 Allocate pseudo RPB
E 13 Allocate pseudo RPB
F 13 Allocate pseudo RPB
G 13 Allocate pseudo RPB
H 13 Allocate pseudo RPB
I 13 Allocate pseudo RPB
J 13 Deallocate RPB
K 13 Check device support tables
L 13 Check device support tables
M 13 Check device support tables
N 13 Check device support tables
B 14 Check device support tables
C 14 Check device support tables
D 14 load error status
E 14 load error status
F 14 load error status
G 14 load XAB
H 14 load XAB
I 14 allocate cache blocks
J 14 allocate cache blocks
K 14 initialize static table
L 14 clean up code

```
M 14  clean up code
N 14  load error code in RAB
B 15  load FAB error code
C 15  verify validity of FAB
D 15  verify cblock
E 15  verify cblock
F 15  verify validity of RAB
G 15  RMS$OPEN routine
H 15  RMS$OPEN routine
I 15  RMS$OPEN routine
J 15  RMS$OPEN routine
K 15  RMS$OPEN routine
L 15  RMS$OPEN routine
M 15  RMS$OPEN routine
N 15  RMS$OPEN routine
B 16  RMS$OPEN routine
C 16  RMS$CONNECT routine
D 16  RMS$CONNECT routine
E 16  RMS$CONNECT routine
F 16  RMS routine
G 16  RMS routine
H 16  RMS routine
I 16  RMS$READ routine
J 16  RMS$READ routine
K 16  RMS$READ routine
L 16  RMS$DISCONNECT routine
```

```
;    1580  2          then
;    1581  2              return rab$error (rms$_ifi);          !  return with error
;    1582  2
;    1583  2          if .cblock [ctl$l_rab] neq .rab           ! If this isn't the right RAB
;    1584  2          then
;    1585  2              return rab$error (rms$_isi);          !  say 'bad stream id'
;    1586  2
;    1587  2          cblock [ctl$l_rab] = 0;                   ! Clear RAB pointer.
;    1588  2          rab$error (rms$_normal)                   ! Set good return
;    1589  1          end;


                                     OFFC 00000          .ENTRY  RMS$DISCONNECT, Save R2,R3,R4,R5,R6,R7,R8,- ; 1525
                                                                 R9,R10,R11                                      ;
                                       59  D4 00002          CLRL    CBLOCK                                  ; 1565
                                       5B  D4 00004          CLRL    FAB                                     ;
                  5A        04         AC  D0 00006          MOVL    RABPTR, RAB                             ; 1566
                                     FBAA  30 0000A          BSBW    VERIFY_RAB                              ; 1568
                  08                   50  E8 0000D          BLBS    R0, 1$                                  ;
                  50 0001863C          8F  D0 00010          MOVL    #99900, R0                              ; 1570
                                           04 00017          RET                                             ;
                            0C         AA  D4 00018 1$:      CLRL    12(RAB)                                 ; 1572
                  5B        3C         AA  D0 0001B          MOVL    60(RAB), FAB                            ; 1573
                                     FB64  30 0001F          BSBW    VERIFY_FAB                              ; 1575
                  09                   50  E8 00022          BLBS    R0, 2$                                  ;
                  50 0001850C          8F  D0 00025          MOVL    #99596, R0                              ; 1577
                                       28  11 0002C          BRB     5$                                      ;
                                     FB68  30 0002E 2$:      BSBW    VERIFY_CBLOCK                           ; 1579
                  09                   50  E8 00031          BLBS    R0, 3$                                  ;
                  50 000'8564          8F  D0 00034          MOVL    #99684, R0                              ; 1581
                                       19  11 0003B          BRB     5$                                      ;
                  5A        20         A9  D1 0003D 3$:      CMPL    32(CBLOCK), RAB                         ; 1583
                                       09  13 00041          BEQL    4$                                      ;
                  50 00018584          8F  D0 00043          MOVL    #99716, R0                              ; 1585
                                       0A  11 0004A          BRB     5$                                      ;
                            20         A9  D4 0004C 4$:      CLRL    32(CBLOCK)                              ; 1587
                  50 00010001          8F  D0 0004F          MOVL    #65537, R0                              ; 1588
                                     FB23  30 00056 5$:      BSBW    RAB$ERROR                               ;
                                           04 00059          RET                                             ; 1589

; Routine Size:  90 bytes,    Routine Base:  CODE + 0825

;    1590  1
```

```
;    1591  1     %sbttl 'RMS$CLOSE routine'
;    1592  1
;    1593  1     global routine rms$close (fabptr) =
;    1594  1
;    1595  1     !++
;    1596  1     ! Functional description
;    1597  1     !       Perform operations necessary to sever a file's connection to
;    1598  1     !       RMS.  This includes de-allocating all internal data blocks,
;    1599  1     !       re-positioning a magtape (if the file is on magtape), and
;    1600  1     !       clearing the file's slot in the RMS$FILE_TABLE.
;    1601  1     !
;    1602  1     ! Inputs:
;    1603  1     !       fabptr          Pointer to the FAB describing the file to be closed
;    1604  1     !
;    1605  1     ! Implicit inputs:
;    1606  1     !       The control block (CBLOCK) associated with the file.
;    1607  1     !
;    1608  1     ! Outputs:
;    1609  1     !       none
;    1610  1     !
;    1611  1     ! Implicit outputs:
;    1612  1     !       FAB [fab$l_sts]         contains return code
;    1613  1     !
;    1614  1     ! Side Effects:
;    1615  1     !       Deallocate file's CBLOCK and clear the file's slot in the file
;    1616  1     !       pointer table, and the FAB's IFI.
;    1617  1     !
;    1618  1     ! Return codes:
;    1619  1     !       RMS$_NORMAL             All OK
;    1620  1     !       possible magtape positioning error code
;    1621  1     !--
;    1622  1
;    1623  2         begin
;    1624  2
;    1625  2         global register
;    1626  2             cblock = 9 : ref bblock ,                   ! map control block
;    1627  2             rab = 10,                                   ! register for rab (not used)
;    1628  2             fab = 11 : ref bblock;                      ! To map FAB
;    1629  2
;    1630  2         local
;    1631  2             stat;
;    1632  2
;    1633  2         cblock = rab = 0;                               ! None
;    1634  2         fab = .fabptr;                                  ! Set up FAB pointer
;    1635  2
;    1636  2         if not verify_fab () then return (rms$_fab);        ! Error if not rea FAB
;    1637  2
;    1638  2         fab [fab$l_sts] = 0;                            ! Init some fields
;    1639  2         fab [fab$l_stv] = 0;
;    1640  2
;    1641  2         if not verify_cblock ()                         ! If bad IFI
;    1642  2         then
;    1643  2             return rms$error (rms$_ifi);                !  return error
;    1644  2
;    1645  2         cblock [ctl$b_op] = ctl$c_close;                ! Specify operation
;    1646  2
;    1647  2         if .cblock [ctl$l_rab] neq 0                    ! If RAB wasn't disconnected
```

```
;    1648  2          then
;    1649  3              begin
;    1650  3
;    1651  3              local
;    1652  3                  stat;
;    1653  3
;    1654  3              stat = rms$disconnect (.cblock [ctl$l_rab]);    ! Disconnect it
;    1655  3
;    1656  3              if not .stat
;    1657  3              then
;    1658  3                  return rms$error (.stat)                    ! Error?
;    1659  2              end;
;    1660  2
;    1661  3          stat = (if .cblock [ctl$l_close] neq 0              ! If there's a close routine
;    1662  3          then (.cblock [ctl$l_close]) ()                     !  call it
;    1663  2          else rms$_normal);                                  !  otherwise, set good status
;    1664  2          rms$error (.stat)                                   ! De-allocate, return
;    1665  1          end;


                                    0FFC 00000          .ENTRY  RMS$CLOSE, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 1593
                                                                R10,R11                                         ;
                                  59  7C 00002           CLRQ    CBLOCK                                      ; 1633
                          5B      04  AC  D0 00004           MOVL    FABPTR, FAB                                 ; 1634
                                    FB21  30 00008           BSBW    VERIFY_FAB                                  ; 1636
                                  08    50  E8 0000B           BLBS    R0, 1$                                      ;
                                  50 0001850C  8F  D0 0000E           MOVL    #99596, R0                          ;
                                             04 00015           RET                                             ;
                                      08  AB  7C 00016 1$:       CLRQ    8(FAB)                                  ; 1638
                                    FB23  30 00019           BSBW    VERIFY_CBLOCK                               ; 1641
                                  09    50  E8 0001C           BLBS    R0, 2$                                      ;
                                  50 00018564  8F  D0 0001F           MOVL    #99684, R0                          ; 1643
                                  25  11 00026           BRB     5$                                              ;
                                  69        04  90 00028 2$:      MOVB    #4, (CBLOCK)                            ; 1645
                                      20  A9  D5 0002B           TSTL    32(CBLOCK)                              ; 1647
                                  0B  13 0002E           BEQL    3$                                              ;
                                      20  A9  DD 00030           PUSHL   32(CBLOCK)                              ; 1654
                        FF6E      CF        01  FB 00033           CALLS   #1, RMS$DISCONNECT                      ;
                                  12    50  E9 00038           BLBC    STAT, 5$                                    ; 1656
                                      14  A9  D5 0003B 3$:      TSTL    20(CBLOCK)                              ; 1661
                                  06  13 0003E           BEQL    4$                                              ;
                            14    B9        00  FB 00040           CALLS   #0, @20(CBLOCK)                         ; 1662
                                  07  11 00044           BRB     5$                                              ;
                                  50 00010001  8F  DC 00046 4$:      MOVL    #65537, STAT                        ; 1661
                                    F95C  30 00049 5$:      BSBW    RMS$ERROR                                   ; 1664
                                          04 00050          RET                                               ; 1665

; Routine Size:  81 bytes,    Routine Base:  CODE + 087F

;    1666  1
;    1667  1      end
;    1668  1
;    1669  0      eludom
```

```
;                                       PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  WORK                           16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  DATA                          688  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                         2256  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                       0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


;                                    Library Statistics
;                                                 -------- Symbols --------       Pages       Processing
;       File                                      Total    Loaded   Percent      Mapped       Time
;
;  DRB1:[DS.WORK]DIAG.L32;265                       784        21         2          85       00:00.2
;  DRB1:[DS.WORK]DS.L32;159                         653        65         9          42       00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7                  18017        97         0         975       00:04.9


;                                          COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE RMS

; Size:          2256 code + 704 data bytes
; Run Time:         00:56.5
; Elapsed Time:     02:48.1
; Lines/CPU Min:     1773
; Lexemes/CPU-Min: 23093
; Memory Used:  274 pages
; Compilation Complete
```

```
;    0001  0      %title '*** RT11 Console media RMS routines'
;    0002  0      module RT11 (                                  ! RMS$OPEN, RMS$GET, RMS$READ, RMS$CLOSE services
;    0003  0                   ident = '01-04'
;    0004  0                   ) =
;    0005  0      !
;    0006  0      ! Copyright (c) 1980, 1981
;    0007  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0008  0      !
;    0009  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0010  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0011  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0012  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0013  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0014  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0015  0      ! REMAIN IN DEC.
;    0016  0      !
;    0017  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0018  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0019  0      ! CORPORATION.
;    0020  0      !
;    0021  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0022  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0023  0      !
;    0024  0      !++
;    0025  0      ! FACILITY: Diagnostic Supervisor
;    0026  0      !
;    0027  0      ! ABSTRACT:
;    0028  0      !
;    0029  0      !       This module supports RMS operations on the VAX console media
;    0030  0      !       (RT-11 formatted RX-01/02 or TU-58).
;    0031  0      !
;    0032  0      ! AUTHOR:
;    0033  0      !       Dave Butenhof,          15-aug-1980
;    0034  0      !
;    0035  0      ! MODIFIED BY:
;    0036  0      !
;    0037  0      !               Dave Butenhof, 6-feb-1981, version 6.3
;    0038  0      !       01      Make $GET return full record size in RAB$L_STV on
;    0039  0      !               RMS$_RTB error.
;    0040  0      !               - Dave butenhof, 02-Jun-1981, version 6.4
;    0041  0      !       02      Fix spec. of libraries for flexibility
;    0042  0      !               - Dave Butenhof, 26-Jun-1981, version 6.4
;    0043  0      !       03      Delete useage of DSP$XX_READ routine; use bootdrivers
;    0044  0      !               (boo$qio) for console storage read.
;    0045  0      !
;    0046  0      !       04      Jack Stansbury, 1-June-1982, Version 6.8
;    0047  0      !               Fixed a truncation error.
;    0048  0      !--
;    0049  0
```

```
;    0050  0     %sbttl 'declarations'
;    0051  1     begin
;    0052  1     !
;    0053  1     ! include files:
;    0054  1     !
;    0055  1
;    0056  1     library '$diag';                !
;    0057  1
;    0058  1     library '$ds';                  !
;    0059  1
;    0060  1     library 'sys$library:lib.l32';
;    0061  1
;    0062  1     !
;    0063  1     ! Internal definitions
;    0064  1     !
;    0065  1
;    0066  1     linkage
;    0067  1         jsb_acc = jsb : global (block = 6, vbn = 7, cblock = 9, rab = 10),
;    0068  1         jsb_adv = jsb (register = 0) : global (vbn = 7, offset = 8);
;    0069  1
;    0070  1     !
;    0071  1     ! table of contents:
;    0072  1     !
;    0073  1
;    0074  1     forward routine
;    0075  1         advance     : Addressing_Mode (Long_Relative) jsb_adv novalue,      ! Advance internal 'RFA' pointers       [04]
;    0076  1         access      : Addressing_Mode (Long_Relative) jsb_acc,              ! Access block of file                  [04]
;    0077  1         rt11$open   : Addressing_Mode (Long_Relative) call_rms,             ! RMS$OPEN emulator--open file          [04]
;    0078  1         rt11$get    : Addressing_Mode (Long_Relative) call_rms,             ! RMS$GET emulator--read/de-block       [04]
;    0079  1         rt11$read   : Addressing_Mode (Long_Relative) call_rms;             ! RMS$READ emulator--read block-mode    [04]
;    0080  1
;    0081  1
;    0082  1     !
;    0083  1     ! External references
;    0084  1     !
;    0085  1
;    0086  1     external routine
;    0087  1         fil$cvtfilnam       : Addressing_Mode (Long_Relative),              ! Convert ascii filename to RAD50
;    0088  1         rms$alloc_cache     : Addressing_Mode (Long_Relative) Jsb_cblock,   ! Allocate cache blocks
;    0089  1         boo$qio             : Addressing_Mode (Long_Relative);              ! Logical block I/O
;    0090  1     !
;    0091  1     ! psect definitions:
;    0092  1     !
;    0093  1
;    0094  1     psect
;    0095  1         plit = data ( share, addressing_mode (long_relative));
;    0096  1
;    0097  1     psect
;    0098  1         own = work ( read, write);
;    0099  1
;    0100  1     psect
;    0101  1         global = work ( read, write);
;    0102  1
;    0103  1     psect
;    0104  1         code = code (share, addressing_mode (long_relative));       !                                           [04]
;    0105  1
```

```
;    0106  1     %sbttl 'advance record pointers'
;    0107  1     routine advance (length) : jsb_adv novalue =
;    0108  2         begin
;    0109  2
;    0110  2         external register
;    0111  2             vbn = 7,
;    0112  2             offset = 8;
;    0113  2
;    0114  2         offset = .offset + .length;                 ! Advance pointer
;    0115  2
;    0116  2         if .offset gtr 511                          ! If we overlapped block
;    0117  2         then
;    0118  3             begin
;    0119  3             vbn = .vbn + (.offset/512);             ! Increment VBN
;    0120  3             offset = .offset and 511                ! Slice down offset
;    0121  3             end
;    0122  3
;    0123  1         end;


                                                                .TITLE  RT11 *** RT11 Console media RMS routines
                                                                .IDENT  \01-04\

                                                                .EXTRN  FIL$CVTFILNAM, RMS$ALLOC_CACHE
                                                                .EXTRN  BOO$QIO

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                58              50  C0 00000 ADVANCE:ADDL2   LENGTH, OFFSET                          ; 0114
                        000001FF 8F             58  D1 00003           CMPL    OFFSET, #511                        ; 0116
                                                10  15 0000A           BLEQ    1$                                  ;
                    50          58 00000200     8F  C7 0000C           DIVL3   #512, OFFSET, R0                    ; 0119
                                57              50  C0 00014           ADDL2   R0, VBN                             ;
            58      58          09              00  EF 00017           EXTZV   #0, #9, OFFSET, OFFSET              ; 0120
                                                    05 0001C 1$:       RSB                                         ; 0123

; Routine Size:  29 bytes,    Routine Base:  CODE + 0000
```

```
;    0124  1     %sbttl 'Access file block via cache'
;    0125  1     routine access : jsb_acc =
;    0126  1
;    0127  1     !++
;    0128  1     ! Functional description:
;    0129  1     !       If the desired VBN is not in cache, refresh cache.  Return
;    0130  1     !       pointer to appropriate cache buffer
;    0131  1     !
;    0132  1     ! Implicit inputs:
;    0133  1     !       R7              desired VBN
;    0134  1     !       R9              RMS control block
;    0135  1     !       R10             RAB
;    0136  1     !
;    0137  1     ! Implicit outputs:
;    0138  1     !       R6              pointer to buffer
;    0139  1     !
;    0140  1     ! Side effects:
;    0141  1     !       may alter cache and perform I/O to floppy.
;    0142  1     !--
;    0143  1
;    0144  2         begin
;    0145  2
;    0146  2         external register
;    0147  2             block = 6,
;    0148  2             vbn = 7,
;    0149  2             cblock = 9 : ref bblock,
;    0150  2             rab = 10 : ref bblock;
;    0151  2
;    0152  2         bind
;    0153  2             cache = cblock [ctl$l_cache1] : vector;
;    0154  2
;    0155  2         if .vbn geq .cblock [ctl$l_eofvbn]          ! If past end of file,
;    0156  2         then
;    0157  2             return (if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof); !  that's it.
;    0158  2
;    0159  2         if .vbn lss .cblock [ctl$l_lvbn]            ! If block's not in
;    0160  2             or .vbn geq .cblock [ctl$l_hvbn]        !  cache now, get it
;    0161  2         then
;    0162  3             begin
;    0163  3
;    0164  3             if .cblock [ctl$w_cache_size] eql 0     ! If no cache buffers
;    0165  3             then
;    0166  4                 begin                               ! make them happen
;    0167  4
;    0168  4                 local
;    0169  4                     stat;
;    0170  4
;    0171  4                 stat = rms$alloc_cache (1);         ! Allocate
;    0172  4
;    0173  4                 if not .stat                        ! If failure
;    0174  4                 then
;    0175  5                     begin
;    0176  5                     rab [rab$l_stv] = .stat;
;    0177  5                     return rms$_dme
;    0178  5                     end
;    0179  5
;    0180  3                 end;
```

```
;    0181  3
;    0182  3             block = .vbn + .cblock [ctl$l_startlbn] - 1;    ! First block to read
;    0183  3
;    0184  3             incr i from 0 to 2 do                          ! Read the blocks
;    0185  4                 begin
;    0186  4
;    0187  4                 local
;    0188  4                     stat;
;    0189  4
;    0190  5                 if (.block - .cblock [ctl$l_startlbn] - 1)  ! If the 3-block window
;    0191  4                     geq .cblock [ctl$l_eofvbn]              !  overlaps EOF, don't
;    0192  4                 then
;    0193  4                     exitloop;                              !  try to read any more
;    0194  4
;    0195  4                 stat = boc$qio (.cache [.i],               ! Read to cache buffer i                    [03]
;    0196  4                     512,                ! one block                                                    [03]
;    0197  4                     .block,             ! logical block number                                         [03]
;    0198  4                     io$_readlblk,       ! read it                                                      [03]
;    0199  4                     0,                  ! physical address                                             [03]
;    0200  4                     .cblock [ctl$l_rpb]);   ! address of RPB                                           [03]
;    0201  4
;    0202  4                 if not .stat
;    0203  4                 then
;    0204  5                     begin
;    0205  5                     rab [rab$l_stv] = .stat;               ! Set read code
;    0206  5                     return rms$_rer
;    0207  4                     end;
;    0208  4
;    0209  4                 block = .block + 1                         ! Advance block #
;    0210  3                 end;
;    0211  3
;    0212  3             cblock [ctl$l_lvbn] = .vbn;                    ! Window begins here
;    0213  3             cblock [ctl$l_hvbn] = min (.vbn + 3,           ! Ends at VBN +3 or
;    0214  3                 .cblock [ctl$l_eofvbn])                    !  at EOF
;    0215  2             end;
;    0216  2
;    0217  2         block = .cache [.vbn - .cblock [ctl$l_lvbn]];       ! Get pointer
;    0218  2         1
;    0219  1         end;                                               ! of access



                                    0C  BB 00000 ACCESS: PUSHR   #^M<R2,R3>                          ; 0125
                          53    2C    A9  9E 00002         MOVAB   44(CBLOCK), R3                    ; 0153
                    44    A9          57  D1 00006         CMPL    VBN, 68(CBLOCK)                   ; 0155
                                      18  19 0000A         BLSS    2$                                ;
                          02    1E    AA  91 0000C         CMPB    30(RAB), #2                       ; 0157
                                      09  12 00010         BNEQ    1$                                ;
                          50 00018650 8F  D0 00012         MOVL    #99932, R0                        ;
                                      70  11 00019         BRB     6$                                ;
                          50 0001827A 8F  D0 0001B 1$:     MOVL    #98938, R0                        ;
                                      67  11 00022         BRB     6$                                ;
                    28    A9          57  D1 00024 2$:     CMPL    VBN, 40(CBLOCK)                   ; 0159
                                      06  19 00028         BLSS    3$                                ;
                    24    A9          57  D1 0002A         CMPL    VBN, 36(CBLOCK)                   ; 0160
```

```
                                                79  19 0002E          BLSS    10$                                          ; 0164
                                          0A    A9  B5 00030 3$:      TSTW    1C(CBLOCK)                                   ;
                                                19  12 00033          BNEQ    4$                                           ; 0171
                                          50    01  D0 00035          MOVL    #1, R0                                       ;
                                          00000000G EF  16 00038       JSB     RMS$ALLOC_CACHE                              ; 0173
                                          0D    50  E8 0003E          BLBS    STAT, 4$                                     ; 0176
                                    0C    AA    50  D0 00041          MOVL    STAT, 12(RAB)                                ; 0177
                                          50 000184D4 8F  D0 00045     MOVL    #99540, R0                                   ;
                                                67  11 0004C          BRB     11$                                          ; 0182
                              50          57    50  A9  C1 0004E 4$:   ADDL3   80(CBLOCK), VBN, R0                         ;
                                          56    FF  A0  9E 00053       MOVAB   -1(R0), BLOCK                                ; 0200
                                                52  D4 00057          CLRL    I                                            ; 0190
                              50          56    50  A9  C3 00059 5$:   SUBL3   80(CBLOCK), BLOCK, R0                       ;
                                                50  D7 0005E          DECL    R0                                           ; 0191
                                    44    A9    50  D1 00060          CMPL    R0, 68(CBLOCK)                               ;
                                                2D  18 00064          BGEQ    8$                                           ; 0200
                                          38    A9  DD 00066          PUSHL   56(CBLOCK)                                   ; 0195
                                          7E        21  7D 00069      MOVQ    #33, -(SP)                                   ; 0197
                                          56        DD  0006C         PUSHL   BLOCK                                        ; 0195
                                          7E  0200  8F  3C 0006E      MOVZWL  #512, -(SP)                                  ;
                                              6342  DD  00073         PUSHL   (R3)[I]                                      ;
                                    00000000G EF        06  FB 00076  CALLS   #6, BOO$QIO                                  ; 0202
                                          0D        50  E8 0007D      BLBS    STAT, 7$                                     ; 0205
                                    0C    AA        50  D0 00080      MOVL    STAT, 12(RAB)                                ; 0206
                                          50 0001C0F4 8F  D0 00084     MOVL    #114932, R0                                  ;
                                                28  11 0008B 6$:      BRB     11$                                          ; 0209
                                                56  D6 0003D 7$:      INCL    BLOCK                                        ;
                              C6          52        02  F3 0008F      AOBLSQ  #2, I, 5$                                    ; 0212
                                    28    A9        57  D0 00093 8$:   MOVL    VBN, 40(CBLOCK)                              ; 0213
                                          50    03  A7  9E 00097       MOVAB   3(R7), R0                                    ; 0214
                                    44    A9        50  D1 0009B      CMPL    R0, 68(CBLOCK)                               ;
                                                04  15 0009F          BLEQ    9$                                           ;
                                          50    44  A9  D0 000A1      MOVL    68(CBLOCK), R0                               ; 0213
                                    24    A9        50  D0 000A5 9$:   MOVL    R0, 36(CBLOCK)                               ; 0217
                              50          57    28  A9  C3 000A9 10$:  SUBL3   40(CBLOCK), VBN, R0                         ;
                                          56        6340  D0 000AE     MOVL    (R3)[R0], BLOCK                              ; 0219
                                          50            01  D0 000B2  MOVL    #1, R0                                       ;
                                                        0C  BA 000B5 11$:  POPR   #^M<R2,R3>                               ;
                                                            05 000B7     RSB                                               ;

; Routine Size:  184 bytes,    Routine Base:  CODE + 001D
```

```
;    0220  1     %sbttl 'RT-11 OPEN service'
;    0221  1
;    0222  1     global routine rt11$open : call_rms =
;    0223  1
;    0224  1     !++
;    0225  1     ! Functional description:
;    0226  1     !       Access a file on the console media (floppy or TU-58) to be
;    0227  1     !       sure it exists and to ascertain it's properties (such as
;    0228  1     !       size).  Return this information in the FAB block and in
;    0229  1     !       the RMS file control block, which is pointed to by the
;    0230  1     !       entry in RMS$FILE_TABLE indexed by the FAB's IFI field.
;    0231  1     !
;    0232  1     ! Inputs:
;    0233  1     !       none
;    0234  1     !
;    0235  1     ! Implicit inputs:
;    0236  1     !       R11     points to the FAB
;    0237  1     !       R9      points to the CBLOCK (RMS internal file control block)
;    0238  1     !
;    0239  1     ! Outputs:
;    0240  1     !       none
;    0241  1     !
;    0242  1     ! Implicit outputs:
;    0243  1     !       Several fields in the FAB and in the static file control block
;    0244  1     !       are altered to describe the file.  Also, if an XAB (XABFHC is
;    0245  1     !       the only XAB currently supported) exists, it will be filled in.
;    0246  1     !
;    0247  1     ! Return status
;    0248  1     !       RMS$_DME        insufficient memory for allocation of buffers
;    0249  1     !       RMS$_NEF        non existant file
;    0250  1     !       RMS$_ACC        can't access media's directory segment
;    0251  1     !--
;    0252  1
;    0253  2         begin
;    0254  2
;    0255  2         local
;    0256  2             ptr,
;    0257  2             len,
;    0258  2             segment,                                ! Directory segment number
;    0259  2             rad50nam : vector [3],                  ! Hold RAD50 name
;    0260  2             rad50desc : bblock [dsc$c_s_bln],       ! Descriptor for RAD50 cvt.
;    0261  2             segblk : bblock [1[.4],                 ! Buffer to read in segment
;    0262  2             address_first,                          ! First entry of segment
;    0263  2             extra_length,                           ! Extra len of entries in seg
;    0264  2             start_block,                            ! Starting block of file
;    0265  2             file_length;                            ! Length of file (blocks)
;    0266  2
;    0267  2         external register
;    0268  2             cblock = 9 : ref bblock,                ! Pointer to control block
;    0269  2             rab = 10,
;    0270  2             fab = 11 : ref bblock;                  ! Define the FAB
;    0271  2
;    0272  2         bind
;    0273  2             unit = cblock [.tl$w_unit] : word;
;    0274  2
;    0275  2         cblock [ctl$l_get] = rt11$get;              ! Store GET routine address
;    0276  2         cblock [ctl$l_read] = rt11$read;            ! Store READ routine address
```

```
;    0277  2          ptr = .cblock [ctl$l_file_ptr];                ! Get filename descriptor
;    0278  2          len = .cblock [ctl$w_file_len];
;    0279  2
;    0280  2          while 1 do                                     ! Skip to end of directory
;    0281  3              begin
;    0282  3
;    0283  3              local
;    0284  3                  char : byte;
;    0285  3
;    0286  3              if (len = .len - 1) lss 0                  ! If no more string
;    0287  3              then
;    0288  3                  return (rms$_fnm);                     !  invalid file name
;    0289  3
;    0290  3              char = ch$rchar_a (ptr);                   ! Get next character
;    0291  3
;    0292  3              if .char eql %ascii']'                     ! If it's a directory
;    0293  3                  or .char eql %ascii'>'                 !  terminator
;    0294  3              then
;    0295  3                  exitloop                               ! then leave scan loop
;    0296  2              end;
;    0297  2
;    0298  2          rad50desc [dsc$w_length] = .len;               ! Load filename descr.
;    0299  2          rad50desc [dsc$a_pointer] = .ptr;
;    0300  2
;    0301  2          if not fil$cvtfilnam (rad50desc, rad50nam)     ! Try to RAD50 the name
;    0302  2          then
;    0303  2              return (rms$_fnm);                         ! Bad filename if can't
;    0304  2
;    0305  2          segment = 1;                                   ! Start with first segment
;    0306  2
;    0307  2          while 1 do                                     ! Find directory entry
;    0308  3              begin
;    0309  3
;    0310  3              local
;    0311  3                  stat;
;    0312  3
;    0313  3              if .segment eql 0 then return (rms$_fnf);      ! If no more, file not found
;    0314  3
;    0315  4              if not (stat = boo$qio (segblk, ! Read to buffer segblk                                  [03]
;    0316  4                          1024,       !  read two blocks (a segment)                                 [03]
;    0317  4                          4 + .segment*2,     !  logical block number                               [03]
;    0318  4                          io$_readlblk,       !  read operation                                     [03]
;    0319  4                          0,          !  physical address                                           [03]
;    0320  4                          .cblock [ctl$l_rpb]))       !                                             [03]
;    0321  3              then
;    0322  4                  (fab [fab$l_stv] = .stat;              ! If can't read, file access
;    0323  3                  return (rms$_acc));                    !  error.
;    0324  3
;    0325  3              segment = .segblk [rt$w_segment];          ! Link to next segment
;    0326  3              start_block = .segblk [rt$w_startblock];       ! Where segment starts
;    0327  3              extra_length = .segblk [rt$w_extralen]; ! Variable part of entries
;    0328  3              address_first = segblk [rt$w_first] - rt$_fixed - .extra_length;        ! First entry address
;    0329  3
;    0330  3              if
;    0331  3
;    0332  3                  while 1 do                             . Find file name in segment
;    0333  4                      begin
```

```
;    0334  4
;    0335  4                    bind
;    0336  4                        entry = (address_first = .address_first + rt$_fixed + .extra length) : bblock;
;    0337  4
;    0338  4                    if .entry [rt$v_endseg] then exitloop 0;        ! End of segment
;    0339  4
;    0340  4                    file_length = .entry [rt$w_filelength]; ! Get length of file
;    0341  4                    start_block = .file_length + .start_block;      ! Start of next file
;    0342  4
;    0343  4                    if .entry [rt$v_perm]                ! If file is permanent
;    0344  4                        and .entry [rt$l_filename] eql .rad50nam [0]        ! and filename matches
;    0345  4                        and .entry [rt$w_filetype] eql .(rad50nam + 6)<0, 16, 0>
;    0346  4                    then
;    0347  4                        exitloop 1
;    0348  4
;    0349  4                    end                                  ! ( value 1 if found, 0 if end of segment )
;    0350  3                then
;    0351  3
;    0352  3                    exitloop                             ! Access file if found
;    0353  2                end;                                     ! of segment loop
;    0354  2
;    0355  2            start_block = .start_block - .file_length;   ! Back up pointer
;    0356  2            !
;    0357  2            ! Initialize control block fields
;    0358  2            !
;    0359  2                cblock [ctl$w_offbyte] = 0;              ! Set fst RFA past EOF
;    0360  2                cblock [ctl$l_eofvbn] = .file_length + 1;!  VBN of said RFA
;    0361  2                cblock [ctl$l_startlbn] = .start_block;  ! First block of file
;    0362  2                fab [fab$l_alq] = .file_length;          ! Set # blocks in file
;    0363  2                fab [fab$l_fop] = fab$m_ctg;             ! RT-11 files are contiguous
;    0364  2                fab [fab$b_fsz] = 0;                     ! Clear fixed record size
;    0365  2                fab [fab$b_rat] = fab$m_cr;              ! Treat as CR format
;    0366  2                fab [fab$b_rfm] = fab$c_rt11;            ! RT-11 format
;    0367  2                fab [fab$w_mrs] = 512;                   ! Max rec size of 1 block
;    0368  2                rms$_normal                              ! Return OK
;    0369  1                end;                                     ! of RT11$OPEN

                                        007C 00000          .ENTRY  RT11$OPEN, Save R2,R3,R4,R5,R6                  ; 0222
                5E      FBEC    CE  9E 00002                MOVAB   -1044(SP), SP                                   ;
          0C    A9 00000000V    EF  9E 00007                MOVAB   RT11$GET, 12(CBLOCK)                            ; 0275
          10    A9 00000000V    EF  9E 0000F                MOVAB   RT11$READ, 16(CBLOCK)                           ; 0276
                51        4C    A9  D0 00017                MOVL    76(CBLOCK), PTR                                 ; 0277
                50        48    A9  3C 0001B                MOVZWL  72(CBLOCK), LEN                                 ; 0278
                                  50  D7 0001F 1$:           DECL    LEN                                             ; 0296
                                  26  19 00021             BLSS    3$                                              ;
                52                81  90 00023             MOVB    (PTR)+, CHAR                                    ; 0290
          5D    8F                52  91 00026             CMPB    CHAR, #93                                       ; 0292
                                  05  13 0002A             BEQL    2$                                              ;
                3E                52  91 0002C             CMPB    CHAR, #62                                       ; 0293
                                  EE  12 0002F             BNEQ    1$                                              ;
          EC    AD                50  B0 00031 2$:        MOVW    LEN, RAD50DESC                                  ; 0298
          F0    AD                51  D0 00035             MOVL    PTR, RAD50DESC+4                                ; 0299
                          F4    AD  9F 00039             PUSHAB  RAD50NAM                                        ; 0301
```

```
                                    EC  AD  9F 0003C          PUSHAB  RAD50DESC                                       ;
                         00000000G  EF      02  FB 0003F          CALLS   #2, FIL$CVTFILNAM                           ;
                                    08      50  E8 00046          BLBS    R0, 4$                                      ;
                                    50 0001852C 8F  D0 00049 3$:       MOVL    #99628, R0                          ; 0303
                                               04 00050          RET                                                 ;
                                    53          01  D0 00051 4$:       MOVL    #1, SEGMENT                         ; 0305
                                                53  D5 00054 5$:       TSTL    SEGMENT                             ; 0313
                                                08  12 00056          BNEQ    6$                                  ;
                                    50 00018292 8F  D0 00058          MOVL    #98962, R0                          ;
                                               04 0005F          RET                                                 ;
                                        38      A9  DD 00060 6$:       PUSHL   56(CBLOCK)                          ; 0320
                                    7E          21  7D 00063          MOVQ    #33, -(SP)                          ; 0315
                              7E    53          01  78 00066          ASHL    #1, SEGMENT, -(SP)                  ; 0317
                                    6E          04  C0 0006A          ADDL2   #4, (SP)                            ;
                                    7E    0400  8F  3C 0006D          MOVZWL  #1024, -(SP)                        ; 0315
                                            14  AE  9F 00072          PUSHAB  SEGBLK                              ;
                         00000000G  EF          06  FB 00075          CALLS   #6, BOO$QIO                         ;
                                    0C      50  E8 0007C          BLBS    STAT, 7$                                    ;
                               0C   AB          50  D0 0007F          MOVL    STAT, 12(FAB)                       ; 0322
                                    50 0001C002 8F  D0 00083          MOVL    #114690, R0                         ; 0323
                                               04 0008A          RET                                                 ;
                                    53    02    AE  3C 0008B 7$:       MOVZWL  SEGBLK+2, SEGMENT                   ; 0325
                                    56    08    AE  3C 0008F          MOVZWL  SEGBLK+8, START_BLOCK               ; 0326
                                    54    06    AE  3C 00093          MOVZWL  SEGBLK+6, EXTRA_LENGTH              ; 0327
                                    50    FC    AE  9E 00097          MOVAB   SEGBLK-4, R0                        ; 0328
                              52    50          54  C3 0009B          SUBL3   EXTRA_LENGTH, R0, ADDRESS_FIRST     ;
                                    52    0E A442 9E  0009F 8$:       MOVAB   14(EXTRA_LENGTH)[ADDRESS_FIRST], -  ; 0336
                                                                              ADDRESS_FIRST                         ;
                                    50          52  D0 000A4          MOVL    ADDRESS_FIRST, R0                   ;
                              A9    60          0B  E0 000A7          BBS     #11, (R0), 5$                       ; 0338
                                    55    08    A0  3C 000AB          MOVZWL  8(R0), FILE_LENGTH                  ; 0340
                                    56          55  C0 000AF          ADDL2   FILE_LENGTH, START_BLOCK            ; 0341
                              E9    60          0A  E1 000B2          BBC     #10, (R0), 8$                       ; 0345
                                    F4  AD    02    A0  D1 000B6          CMPL    2(R0), RAD50NAM                     ; 0344
                                                    E2  12 000BB          BNEQ    8$                                  ;
                                    FA  AD    06    A0  B1 000BD          CMPW    6(R0), RAD50NAM+6                   ; 0345
                                                    0B  12 000C2          BNEQ    8$                                  ;
                                    56          55  C2 000C4          SUBL2   FILE_LENGTH, START_BLOCK            ; 0355
                                            42  A9  B4 000C7          CLRW    66(CBLOCK)                          ; 0359
                                    44  A9    01    A5  9E 000CA          MOVAB   1(R5), 68(CBLOCK)                   ; 0360
                                    50  A9          56  D0 000CF          MOVL    START_BLOCK, 80(CBLOCK)             ; 0361
                                    10  AB          55  D0 000D3          MOVL    FILE_LENGTH, 16(FAB)                ; 0362
                                    04  AB 00100000 8F  D0 000D7          MOVL    #1048576, 4(FAB)                    ; 0363
                                              3F    AB  94 000DF          CLRB    63(FAB)                             ; 0364
                                    1E  AB      0402  8F  B0 000E2          MOVW    #1026, 30(FAB)                      ; 0365
                                    36  AB      0200  8F  B0 000E8          MOVW    #512, 54(FAB)                       ; 0367
                                    50 00010001 8F  D0 000EE          MOVL    #65537, R0                          ; 0369
                                               04 000F5          RET                                                 ;

; Routine Size:  246 bytes,    Routine Base:  CODE + 00D5

;   0370  ;
```

```
;    0371  1      %sbttl 'RT11$GET routine'
;    0372  1
;    0373  1      global routine rt11$get : call_rms =
;    0374  1
;    0375  1      !++
;    0376  1      ! Functional description:
;    0377  1      !       Access a single record from the record stream, copying it to
;    0378  1      !       the user's buffer.
;    0379  1      !
;    0380  1      ! Inputs:
;    0381  1      !       none
;    0382  1      !
;    0383  1      ! Implicit inputs:
;    0384  1      !       R9              pointer to internal control CBLOCK
;    0385  1      !       R10             pointer to RAB
;    0386  1      !       various fields within all three control structures
;    0387  1      !
;    0388  1      ! Outputs:
;    0389  1      !       nore
;    0390  1      !
;    0391  1      ! Implicit outputs:
;    0392  1      !       RAB [rab$l_rbf] Address where record is stored
;    0393  1      !       RAB [rab$w_rsz] Size of record
;    0394  1      !       RAB [rab$l_sts] Status return code
;    0395  1      !
;    0396  1      ! Side effects:
;    0397  1      !       CBLOCK          current RFA will be updated if access is sequential.
;    0398  1      !                       record size is set if read is successful.
;    0399  1      !                       Cache contents and control data may be affected
;    0400  1      ! Status codes:
;    0401  1      !       RMS$_NORMAL     good return
;    0402  1      !       RMS$_RTB        record too big (truncated)
;    0403  1      !       RMS$_EOF        RFA is beyond last record of file
;    0404  1      !       RMS$_RER        Read error on media
;    0405  1      !--
;    0406  1
;    0407  2          begin
;    0408  2
;    0409  2          external register
;    0410  2              cblock = 9 : ref bblock,                    ! Map CBLOCK
;    0411  2              rab = 10 : ref bblock;                      ! Map RAB
;    0412  2
;    0413  2          global register
;    0414  2              block = 6 : ref vector [, byte],            ! Access block
;    0415  2              vbn = 7,                                    ! Current file 'VBN'
;    0416  2              offset = 8;                                 ! Current block offset
;    0417  2
;    0418  2          local
;    0419  2              outlim,                                     ! Remaining length of buffer
;    0420  2              outptr,                                     ! Address to copy to
;    0421  2              rtb;                                        ! Flag for record truncated
;    0422  2
;    0423  2          vbn = .cblock [ctl$l_vbn];                      ! Grab the intended RFA
;    0424  2          offset = .cblock [ctl$w_byte];
;    0425  2
;    0426  2          if .rab [rab$b_rac] neq rab$c_rta               ! If not random access
;    0427  2          then
```

```
;    0428  2              advance (.cblock [ctl$w_rec_size]);     ! advance RFA to next record
;    0429  2
;    0430  2          cblock [ctl$l_vbn] = .vbn;                  ! Save new RFA
;    0431  2          cblock [ctl$w_byte] = .offset;
;    0432  2          cblock [ctl$w_rec_size] = 0;                ! If bad error, repeat record
;    0433  2          rtb = 0;                                    ! Record's not too big yet
;    0434  2          rab [rab$w_rsz] = 0;                        ! Clear record size
;    0435  2          outlim = .rab [rab$w_usz];                  ! Limiting size for transfer
;    0436  2          rab [rab$l_rbf] = outptr = .rab [rab$l_ubf];        ! Set output pointer
;    0437  2
;    0438  2          while 1 do                                  ! Read record
;    0439  3              begin
;    0440  3
;    0441  3              while 1 do                              ! Read up to CR or EOF
;    0442  4                  begin
;    0443  4
;    0444  4                  local
;    0445  4                      cr,                             ! Flag
;    0446  4                      string_size,                    ! Size of partial string
;    0447  4                      bite,                           ! Current offset in block
;    0448  4                      len;                            ! Length to copy
;    0449  4
;    0450  4                  cr = 0;                             ! Clear CR flag
;    0451  4                  bite = .offset;                     ! Where to start in block
;    0452  5                  begin
;    0453  5
;    0454  5                  local
;    0455  5                      stat;
;    0456  5
;    0457  5                  stat = access ();                   ! Access the block
;    0458  5
;    0459  5                  if not .stat then return .stat
;    0460  5
;    0461  4                  end;
;    0462  4
;    0463  4                  if .block [.offset] eql 0           ! If reached EOF
;    0464  4                  then
;    0465  4                      return (if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof); !  get out
;    0466  4
;    0467  5                  while not (cr = .block [.bite] eql %x'd')   ! Find end of block or CR
;    0468  4                      and .bite leq 511 do
;    0469  4                      bite = .bite + 1;               ! Advance
;    0470  4
;    0471  4                  string_size = .bite - .offset;      ! Get length of this string
;    0472  4                  len = min (.string_size, .outlim);  ! Get length to transfer
;    0473  4
;    0474  4                  if .outlim lss .string_size         ! If not enough space,
;    0475  4                  then
;    0476  4                      rtb = 1;                        !  set flag for overflow
;    0477  4
;    0478  4                  outptr = ch$move (.len, block [.offset], .outptr);  ! Copy as much as we can
;    0479  4                  outlim = .outlim - .len;            ! Count size of copy
;    0480  4                  rab [rab$w_rsz] = .rab [rab$w_rsz] + .len;
;    0481  4                  advance (.string_size);             ! Advance past this block
;    0482  4
;    0483  4                  if .cr
;    0484  4                  then
```

```
;    0485  4                    exitloop                        ! Keep looping until end
;    0486  3                end;
;    0487  3
;    0488  3            advance (1);                            ! Skip the CR
;    0489  4            begin
;    0490  4
;    0491  4            local
;    0492  4                stat;
;    0493  4
;    0494  4            stat = access ();                       ! Access new RFA
;    0495  4
;    0496  4            if not .stat then return .stat
;    0497  4
;    0498  3            end;
;    0499  3
;    0500  3            if .block [.offset] eql %x'a'           ! If CR is followed by LF
;    0501  3            then
;    0502  4                begin
;    0503  4                cblock [ctl$w_rec_size] =           ! Set whole record size
;    0504  4                (.vbn - .cblock [ctl$l_vbn])*512 + .offset - .cblock [ctl$w_byte] + 1;
;    0505  4                return
;    0506  5                    (if .rtb
;    0507  5                    then
;    0508  6                        begin
;    0509  6                        rab [rab$l_stv] = .cblock [ctl$w_rec_size];
;    0510  6                        rms$_rtb
;    0511  6                        end
;    0512  5                    else rms$_normal)
;    0513  4                end
;    0514  3            else                                    ! Not end of record yet
;    0515  3
;    0516  3                if .outlim lss 1                    : If there's no room
;    0517  3                then
;    0518  3                    rtb = 1
;    0519  3                else
;    0520  4                    begin
;    0521  4                    outlim = .outlim - 1;           ! Insert the CR we found
;    0522  4                    rab [rab$w_rsz] = .rab [rab$w_rsz] + 1;
;    0523  4                    ch$wchar_a (%x'd', outptr)
;    0524  4                    end
;    0525  4
;    0526  3                end
;    0527  3
;    0528  1        end;                                        ! of RT11$GET



                                    09FC 00000          .ENTRY  RT11$GET, Save R2,R3,R4,R5,R6,R7,R8,R11     ; 0373
                                5E      10  C2 00002    SUBL2   #16, SP                                     ;
                                57      18  A9 D0 00005 MOVL    24(CBLOCK), VBN                             ; 0423
                                58      1C  A9 3C 00009 MOVZWL  28(CBLOCK), OFFSET                          ; 0424
                                02      1E  AA 91 0000D CMPB    30(RAB), #2                                 ; 0426
                                            08 13 00011 BEQL    1$                                          ;
                                50      1E  A9 3C 00013 MOVZWL  30(CBLOCK), R0                              ; 0428
                                    FE1A    CF 16 00017 JSB     ADVANCE                                     ;
```

```
                                18   A9              57   D0 0001B 1$:     MOVL    VBN, 24(CBLOCK)                          ; 0430
                                1C   A9              58   3C 0001F         MOVZWL  OFFSET, 28(CBLOCK)                       ; 0431
                                              08     AE   D4 00023         CLRL    RTB                                      ; 0433
                                              22     AA   B4 00026         CLRW    34(RAB)                                  ; 0434
                                04   AE          20   AA   3C 00029         MOVZWL  32(RAB), OUTLIM                          ; 0435
                                53                24   AA   D0 0002E         MOVL    36(RAB), OUTPTR                          ; 0436
                                28   AA                   53   D0 00032         MOVL    OUTPTR, 40(RAB)                          ;
                                                    0C     AE   D4 00036 2$:     CLRL    CR                                       ; 0450
                                52                          58   D0 00039         MOVL    OFFSET, BITE                             ; 0451
                                                  FE12      CF   16 0003C         JSB     ACCESS                                   ; 0457
                                01                          50   E8 00040         BLBS    STAT, 3$                                 ; 0459
                                                            04   00043         RET                                              ;
                                                       6846 95   00044 3$:     TSTB    (OFFSET)[BLOCK]                          ; 0463
                                                         16   12   00047         BNEQ    5$                                       ;
                                02                  1E   AA   91   00049         CMPB    30(RAB), #2                              ; 0465
                                                         08   12   0004D         BNEQ    4$                                       ;
                                50 0001865C              8F   D0   0004F         MOVL    #99932, R0                               ;
                                                            04   00056         RET                                              ;
                                50 0001827A              8F   D0   00057 4$:     MOVL    #98938, R0                               ;
                                                            04   0005E         RET                                              ;
                                                            50   D4   0005F 5$:     CLRL    R0                                       ; 0467
                                0D                       6246 91   00061         CMPB    (BITE)[BLOCK], #13                       ;
                                                            02   12   00065         BNEQ    6$                                       ;
                                                            50   D6   00067         INCL    R0                                       ;
                                0C   AE                     50   D0   00069 6$:     MOVL    R0, CR                                   ;
                                0D                          50   E8   0006D         BLBS    R0, 7$                                   ;
                          000001FF   8F                     52   D1   00070         CMPL    BITE, #511                               ; 0468
                                                            04   14   00077         BGTR    7$                                       ;
                                                            52   D6   00079         INCL    BITE                                     ; 0469
                                                            E2   11   0007B         BRB     5$                                       ;
                             6E                 52          58   C3   0007D 7$:     SUBL3   OFFSET, BITE, STRING_SIZE                ; 0471
                                                50          6E   D0   00081         MOVL    STRING_SIZE, R0                          ; 0472
                                          04    AE          50   D1   00084         CMPL    R0, OUTLIM                               ;
                                                            04   15   00088         BLEQ    8$                                       ;
                                                50    04    AE   D0   0008A         MOVL    OUTLIM, R0                               ;
                                                5B          50   D0   0008E 8$:     MOVL    R0, LEN                                  ;
                                                6E    04    AE   D1   00091         CMPL    OUTLIM, STRING_SIZE                      ; 0474
                                                            04   18   00095         BGEQ    9$                                       ;
                                          08    AE          01   D0   00097         MOVL    #1, RTB                                  ; 0476
                             63                   6846        5B   28   0009B 9$:     MOVC3   LEN, (OFFSET)[BLOCK], (OUTPTR)           ; 0478
                                          04    AE          5B   C2   000A0         SUBL2   LEN, OUTLIM                              ; 0479
                                          22    AA          5B   A0   000A4         ADDW2   LEN, 34(RAB)                             ; 0480
                                                50          6E   D0   000A8         MOVL    STRING_SIZE, R0                          ; 0481
                                                      FD86  CF   16   000AB         JSB     ADVANCE                                  ;
                                                59    0C    AE   E9   000AF         BLBC    CR, 13$                                  ; 0483
                                                50          01   D0   000B3         MOVL    #1, R0                                   ; 0488
                                                      FD7B  CF   16   000B6         JSB     ADVANCE                                  ;
                                                      FD94  CF   16   000BA         JSB     ACCESS                                   ; 0494
                                                4E          50   E9   000BE         BLBC    STAT, 14$                                ; 0496
                                                0A                6846 91   000C1         CMPB    (OFFSET)[BLOCK], #10                     ; 0500
                                                            31   12   000C5         BNEQ    11$                                      ;
                             50                 57    18    A9   C3   000C7         SUBL3   24(CBLOCK), VBN, R0                      ; 0504
                             50                 50          09   78   000CC         ASHL    #9, R0, R0                               ;
                                                50          58   C0   000D0         ADDL2   OFFSET, R0                               ;
                                                51    1C    A9   3C   000D3         MOVZWL  28(CBLOCK), R1                           ;
                                                50          51   C2   000D7         SUBL2   R1, R0                                   ;
                          1E    A9                50          01   A1   000DA         ADDW3   #1, R0, 30(CBLOCK)                       ;
```

```
                                    0D        08    AE  E9 000DF          BLBC    RTB, 10$                                    ; 0506
                                OC  AA        1E    A9  3C 000E3          MOVZWL  3C(CBLOCK), 12(RAB)                         ; 0509
                                    50 000181A8     8F  D0 000E8          MOVL    #98728, R0                                  ; 0508
                                                    04  000EF          RET                                                 ;
                                    50 00010001     8F  D0 000F0 10$:     MOVL    #65537, R0                                  ; 0506
                                                    04  000F7          RET                                                 ;
                                             04     AE  D5 000F8 11$:     TSTL    OUTLIM                                      ; 0516
                                                    06  14 000FB          BGTR    12$                                         ;
                                OB  AE              01  D0 000FD          MOVL    #1, RTB                                     ; 0518
                                                    09  11 00101          BRB     13$                                         ;
                                             04     AE  D7 00103 12$:     DECL    OUTLIM                                      ; 0521
                                             22     AA  B6 00106          INCW    34(RAB)                                     ; 0522
                                    83              0D  90 00109          MOVB    #13, (OUTPTR)+                              ; 0523
                                                FF27    31 0010C 13$:     BRW     2$                                          ; 0500
                                                    04  0010F 14$:     RET                                                 ; 0528

; Routine Size:  272 bytes,    Routine Base:  CODE + 01CB

;   0529  1
```

```
;    0530  1      %sbttl 'RT11$READ routine'
;    0531  1
;    0532  1      global routine rt11$read : call_rms =
;    0533  1
;    0534  1      !++
;    0535  1      ! Functional description:
;    0536  1      !       Read virtual blocks of file
;    0537  1      !
;    0538  1      ! Inputs:
;    0539  1      !       none
;    0540  1      !
;    0541  1      ! Implicit inputs:
;    0542  1      !       R9                      CBLOCK pointer
;    0543  1      !       R10                     RAB pointer
;    0544  1      !       various fields in the above-listed data structures
;    0545  1      !
;    0546  1      ! Outputs:
;    0547  1      !       none
;    0548  1      !
;    0549  1      ! Implicit outputs:
;    0550  1      !       RAB [rab$l_rbf]         points to buffer
;    0551  1      !       RAB [rab$w_rsz]         size of buffer
;    0552  1      !
;    0553  1      ! Side Effects:
;    0554  1      !       CBLOCK [ctl$l_nbp]      points to next VBN for sequential READ.
;    0555  1      !
;    0556  1      ! Return codes:
;    0557  1      !       RMS$_NORMAL             OK
;    0558  1      !       RMS$_RER                Read error
;    0559  1      !--
;    0560  1
;    0561  2          begin
;    0562  2
;    0563  2          external register
;    0564  2              cblock = 9 : ref bblock,                ! Map to control block
;    0565  2              rab = 10 : ref bblock;                  ! Map to RAB
;    0566  2
;    0567  2          local
;    0568  2              size,
;    0569  2              stat,
;    0570  2              next;
;    0571  2
;    0572  2          next = .cblock [ctl$l_nbp];                 ! VBN to read
;    0573  2          rab [rab$w_rsz] = size = min (.rab [rab$w_usz],     ! smaller of given size
;    0574  2              (.cblock [ctl$l_eofvbn] - .next)*512);  ! and rest of file size
;    0575  2          cblock [ctl$l_nbp] = .next + (.size + 511)/512;     ! Set next block pointer
;    0576  2          stat = boo$qio (                ! Read the virtual block                        [03]
;    0577  2                          rab [rab$l_rbf] = .rab [rab$l_ubf], !  to specified place        [03]
;    0578  2                          .size,          !  with specified size                          [03]
;    0579  2                          .cblock [ctl$l_startlbn] + .next - 1,       ! logical          [03]
;    0580  2                                                                      ! block            [03]
;    0581  2                          io$_readlblk,       !  logical read                             [03]
;    0582  2                          0,              !  physical addresses                           [03]
;    0583  2                          .cblock [ctl$l_rpb]);       !                                   [03]
;    0584  2
;    0585  2          if not .stat
;    0586  2          then
```

```
;    0587  3            begin
;    0588  3            rab [rab$l_stv] = .stat;                ! Save 'system' return
;    0589  3            return rms$_rer                         ! Return read error
;    0590  2            end;
;    0591  2
;    0592  2        rms$_normal
;    0593  1        end;


                                                0004 00000          .ENTRY  RT11$READ, Save R2                          ; 0532
                                  52       04   A9   D0 00002          MOVL    4(CBLOCK), NEXT                     ; 0572
                            50    44   A9        52   C3 00006          SUBL3   NEXT, 68(CBLOCK), R0                ; 0574
                            50              50   09   78 0000B          ASHL    #9, R0, R0                          ;
                                            51   AA   3C 0000F          MOVZWL  32(RAB), R1                         ;
                                            50   51   D1 00013          CMPL    R1, R0                              ;
                                                 03   15 00016          BLEQ    1$                                  ;
                                            51        50   D0 00018          MOVL    R0, R1                          ;
                                            50        51   D0 0001B 1$:      MOVL    R1, SIZE                        ; 0573
                                       22   AA        51   B0 0001E          MOVW    R1, 34(RAB)                     ;
                                            51  01FF   C0   9E 00022          MOVAB   511(R0), R1                     ; 0575
                                            51 00000200 8F   C6 00027          DIVL2   #512, R1                        ;
                                  04   A9        51        52   C1 0002E          ADDL3   NEXT, R1, 4(CBLOCK)         ;
                                                 38   A9   DD 00033          PUSHL   56(CBLOCK)                      ; 0583
                                            7E        21   7D 00036          MOVQ    #33, -(SP)                      ; 0576
                                  51             52   50   A9   C1 00039          ADDL3   80(CBLOCK), NEXT, R1        ; 0579
                                                 FF   A1   9F 0003E          PUSHAB  -1(R1)                          ;
                                                      50   DD 00041          PUSHL   SIZE                            ; 0578
                                                 24   AA   DD 00043          PUSHL   36(RAB)                         ; 0577
                                       28   AA             6E   D0 00046          MOVL    (SP), 40(RAB)           ;
                                 00000000G  EF             06   FB 0004A          CALLS   #6, B00$QIO             ;
                                            0C             50   E8 00051          BLBS    STAT, 2$                ; 0585
                                       0C   AA             50   D0 00054          MOVL    STAT, 12(RAB)           ; 0588
                                            50 0001C0F4    8F   D0 00058          MOVL    #114932, R0             ; 0589
                                                           04   0005F          RET                                 ;
                                            50 00010001    8F   D0 00060 2$:      MOVL    #65537, R0              ; 0593
                                                           04   00067          RET                                 ;

; Routine Size:  104 bytes,    Routine Base:  CODE + 02DB

;    0594  1
;    0595  1        end
;    0596  1
;    0597  0        eludom


;                                   PSECT SUMMARY
;
;        Name                    Bytes                           Attributes
;
```

```
;  CODE                              835  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                                 Library Statistics
;
;                                          --------- Symbols ---------      Pages       Processing
;       File                               Total     Loaded   Percent      Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265                784          0         0          85        00:00.2
;  DRB1:[DS.WORK]DS.L32;159                  653         31         4          42        00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7           18017         29         0         975        00:04.7



;                                      COMMAND QUALIFIERS
;
;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE RT11
;
; Size:          835 code + 0 data bytes
; Run Time:          00:20.8
; Elapsed Time:      00:44.1
; Lines/CPU Min:      1718
; Lexemes/CPU-Min:   12584
; Memory Used:   132 pages
; Compilation Complete
```

Table of contents

(1) 124 DECLARATIONS
(1) 183 INITIAL SYSTEM CONTROL BLOCK IMAGE.
(1) 256 DSX$INITSCB INITIALIZE SYSTEM CONTROL BLOCK ROUTINE
(1) 352 DSX$SETVEC SET VECTOR ROUTINE
(1) 471 DSX$CLRVEC RESTORE VECTOR ROUTINE
(1) 549 CHECK_VECTOR Verify vector range
(1) 594 DSX$SETIPL SET INTERRUPT PRIORITY LEVEL SERVICE PROCEDURE.
(1) 627 SOFT$INT Software interrupt vectors

```
                        0000     1          .TITLE  SCB     SYSTEM CONTROL BLOCK
                        0000     2          .IDENT  /07-29/
                        0000     3          .NoShow Conditionals
                        0000     4
                        0000     5 ;
                        0000     6 ; COPYRIGHT (C) 1977, 1981, 1983
                        0000     7 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000     8 ;
                        0000     9 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    10 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    11 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    12 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    13 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    14 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    15 ; REMAIN IN DEC.
                        0000    16 ;
                        0000    17 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    18 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    19 ; CORPORATION.
                        0000    20 ;
                        0000    21 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    22 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    23
                        0000    24 ;++
                        0000    25 ; FACILITY:
                        0000    26 ;
                        0000    27 ; ABSTRACT:
                        0000    28 ;
                        0000    29 ; ENVIRONMENT:
                        0000    30 ;
                        0000    31 ; AUTHOR:        KEN CHAPMAN     20-SEP-77       VERSION 01.
                        0000    32 ;
                        0000    33 ; MODIFIED BY:
                        0000    34 ;                NICK HOWGATE    05-DEC-77       VERSION 02
                        0000    35 ;        01      ADDED EXCEPTION HANDLER ENHANCEMENT.
                        0000    36 ;
                        0000    37 ;                KEN CHAPMAN     01-DEC-77       VERSION 03
                        0000    38 ;        02      ADD TIMRON FLAG SET AND CLEAR TO $DS_SETVEC_L AND $DS_CLRVEC_L,
                        0000    39 ;                RESPECTIVELY, FOR INTERVAL TIMER VECTOR.
                        0000    40 ;
                        0000    41 ;                KEN CHAPMAN     16-FEB-78       VERSION 04 (ESSAA-3.07).
                        0000    42 ;        03      GENERAL CODE ENHANCEMENTS.
                        0000    43 ;        04      ADDED GENERALIZED EXCEPTION HANDLER FROM DEBUG.
                        0000    44 ;        05      "REI" INSTEAD OF "HALT" ON UNEXPECTED INTERRUPT.
                        0000    45 ;
                        0000    46 ;                N. HOWGATE      21-FEB-78       VERSION 05 (ESSAA-4.00).
                        0000    47 ;        06      ADD QIO SUPPORT - SOFTWARE LEVEL 8 VECTOR, AND MBA0 VECTOR
                        0000    48 ;                TOM SOUTTER     27-FEB-78
                        0000    49 ;        07      FIX TO UNEXPECTED VECTOR HANDLER TO CALL CLI
                        0000    50 ;                TOM SOUTTER     28-FEB-78
                        0000    51 ;        08      TURNED ON AST DELIVERY VECTOR (SOFTWARE LEVEL 2 INTERRUPT)
                        0000    52 ;
                        0000    53 ;                TOM SOUTTER     28-MAR-78       VERSION 06 (ESSAA-4.01).
                        0000    54 ;        09      INSTALLED HOOK TO INITIALIZE SYSTEM TIME IN $DS_CLRVEC_L (TIMER)
                        0000    55 ;        10      SET TIMER VECTOR TO DSI$TIMSRV & TR9/BR5 TO MBA$INTDIS
                        0000    56 ;                ROGER RIGGS     20-APR-78
                        0000    57 ;        11      CORRECTLY REMOVE UPPER 2 BITS OF SBA ADDRESS ON DISPLAY
```

```
                        0000     58 ;        12      Get PC correct and do DMP_SBI and DMP_SILO on SBI errors
                        0000     59 ;                ROGER RIGGS     15-FEB-79
                        0000     60 ;        13      MODIFIED SETVEC SO LOW 2 BITS OF SCB REFLECT SETVEC BITS
                        0000     61 ;                EVEN FOR 'SOFT' VECTORS.
                        0000     62 ;                REMOVED 'IS' BIT FROM MOST EXCEPTIONS
                        0000     63 ;                Roger Riggs     12-Jun-1979
                        0000     64 ;        14      Made software interrupt vectors soft, like BPT, T-bit...
                        0000     65 ;                Roger Riggs     1-OCT-1979
                        0000     66 ;        15      Changes to have only 1 SCB instead of 1 per machine
                        0000     67 ;                Space for SCB is allocated in kernel, initialization
                        0000     68 ;                is done here.  SCB_IMAGE is reduced to only 1 page.
                        0000     69 ;                Roger Riggs     6-NOV-1979
                        0000     70 ;        16      Made SETIPL change mode to KERNEL before setting the IPL
                        0000     71 ;                Roger Riggs, 26-Jan-1980, Version 5.2
                        0000     72 ;        17      Set supervisor vector for software IPL interrupts on
                        0000     73 ;                levels 6,9,10,11, to dispatch to EXE$FORKDSPTH
                        0000     74 ;                John Ciukaj, 5-feb-80.    Version 5.2
                        0000     75 ;        18      Delete 'MOVAB   IS(R3),(R2)' causing exception while
                        0000     76 ;                running diagnostic ECKAX, Exception depended on value
                        0000     77 ;                of R2 on entry to DS$INITSCB
                        0000     78 ;                Roger Riggs, 21-Mar-1980
                        0000     79 ;        19      Added EXE$GL_SCB to hold address of current SCB.
                        0000     80 ;                Roger Riggs, 18-SEP-1980, Version 6.0
                        0000     81 ;        20      Added code to DS$INITSCB to restart the clock
                        0000     82 ;                Dave Butenhof, 18-feb-1981, version 6.3
                        0000     83 ;        21      Also allow SETVEC w/out changing address
                        0000     84 ;        22      Fix several truncation errors
                        0000     85 ;
                        0000     86 ;        23      - Jack Stansbury, 22-Oct-1981, Version 6.5
                        0000     87 ;                Fixed several truncation errors. Changed SEP Psects to
                        0000     88 ;                what they should be. Added .LIBRARY statements for
                        0000     89 ;                $DS and $DIAG.
                        0000     90 ;
                        0000     91 ;        24      Jack Stansbury, 15-September-1982, Version 6.9
                        0000     92 ;                Added code in the SetVec routine that will take care of a race
                        0000     93 ;                condition with the interval clock interrupts and an MTPR
                        0000     94 ;                instruction that turns off the clock. Also changed the ICCS$M_ERR
                        0000     95 ;                to be bit 15, instead of bit 31.
                        0000     96 ;
                        0000     97 ;        25      Jack Stansbury, 16-September-1982, Version 6.9
                        0000     98 ;                Changed the ICCS$M_Err bit back to 31. The VAX Hardware Handbook
                        0000     99 ;                has a picture of the ICCS longword in the back of the book. This
                        0000    100 ;                picture states that the ERR bit is bit 15. However, in other
                        0000    101 ;                sections of the book (and in the SRM), the ERR bit is 15. The
                        0000    102 ;                NEBULA hardware thinks it is 15 also.
                        0000    103 ;
                        0000    104 ;        26      Bob Bergazzi    21-Oct-1982     version 6.9
                        0000    105 ;                Changed comments in DSX$SETVEC routine. Also see CLOCK.MAR [25]
                        0000    106 ;
                        0000    107 ;        27      Bob Bergazzi    April 26, 1983  Version 6.11
                        0000    108 ;                Added call to ONLY_SCB in DSX$INITSCB which fixes the SCB for XXX.
                        0000    109 ;
                        0000    110 ;        28      Bob Bergazzi    June 7, 1983    Version 6.12
                        0000    111 ;                Changed the CLRVEC routine to reset the clock vector before
                        0000    112 ;                resetting the clock. This compensates for the TOK hardware bug
                        0000    113 ;                on 11/750s, which caused interrupts once in a blue moon when
                        0000    114 ;                resetting the clock.
```

```
                        0000   115 ;
                        0000   116 ;       29      Domenic Andella 1-May-1984      Version 7.0
                        0000   117 ;               Included FAULT_CLEAR selection code macro, and added
                        0000   118 ;               push/pop of selection code around call to DSR$FAULT_CLEAR
                        0000   119 ;               in routine DSX$INITSCB. Specific action relating to
                        0000   120 ;               CPU type and calling routine can then be taken in
                        0000   121 ;               MCHK___ module.
                        0000   122 ;--
```

```
                                0000   124          .SBTTL  DECLARATIONS
                                0000   125 ;
                                0000   126 ; INCLUDE FILES:
                                0000   127 ;
                                0000   128          .LIBRARY        /SYS$LIBRARY:LIB/       ;                       [23]
                                0000   129          .LIBRARY        /$DS/           ;                               [23]
                                0000   130          .LIBRARY        /$DIAG/         ;                               [23]
                                0000   131
                                0000   132 ;
                                0000   133 ; MACROS:
                                0000   134 ;
                                0000   135
                                0000   136 ;
                                0000   137 ; EQUATED SYMBOLS:
                                C000   138 ;
                                0000   139
                                0000   140          $DS_CVTREG_DEF
                                0000   141          $DS_DSDEF
                                0000   142          $PRDEF
                                0000   143          $PSLDEF
                                0000   144          $DS_SCBDEF
                                0000   145          $DS_SETVEC_DEF
                                0000   146          CMKDEF
                                0000   147          DSFDEF
                                0000   148          FLTCLR_SEL                                                     ;[29
                    00000001    0000   149 IS       = 1
                                0000   150
                    80000000    0000   151 ICCS$M_ERR       = 1 @ 31                ; ERROR INDICATOR IN PR$_ICCS      [25]
                    00000080    0000   152 ICCS$M_INT       = 1 @ 7                 ; INTERRUPT PENDING IN $PR$_ICCS
```

```
                                00000000   154          .PSECT  Work, NoExe, NoShr, Wrt, Long   ;                                [23]
                                    0000   155
                                    0000   156 ;
                                    0000   157 ; OWN STORAGE:
                                    0000   158 ;
                                    0000   159
                                    0000   160 DS$GB_Stopping_The_Timer::      ; If = 1, we are doing an MTPR to stop the       [24]
                                00  0000   161          .Byte   0              ; . . . interval clock.                          [24]
                                    0001   162
                                    0001   163 DS$GA_CHMKVEC::
                           00000005 0001   164          .BLKL   1              ; USER CHANGE MODE TO KERNEL VECTOR.
                                    0005   165
                                    0005   166 DS$GA_BREAKVEC::
                           00000009 0005   167          .BLKL   1              ; USER BREAK INSTRUCTION FAULT VECTOR.
                                    0009   168
                                    0009   169 DS$GA_TBITVEC::
                           0000000D 0009   170          .BLKL   1              ; USER T-BIT TRAP VECTOR.
                                    000D   171
                                    000D   172 EXE$GL_SCB::
                           00000000' 000D  173          .LONG   SCB_IMAGE      ; Virtual address of SCB
                                    0011   174
                                    0011   175 DS$GA_SOFTINT::
                           00000000 0011   176          .LONG   0              ; 1 Bit per level 1-15(D)
00000000'00000000'00000000'00000000' 0015   177          .LONG   0[15]          ; Soft interrupt vectors, 0=not assigned
00000000'00000000'00000000'00000000' 0025
00000000'00000000'00000000'00000000' 0035
          00000000'00000000'00000000' 0045
```

```
                        00000000    179         .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                                   [23]
                            0000    180
                            0000    181         MODNAM  SCB
```

```
                                0004    183          .SBTTL  INITIAL SYSTEM CONTROL BLOCK IMAGE.
                                0004    184 ;+
                                0004    185 ; IMAGE OF THE SYSTEM CONTROL BLOCK USED TO REINITIALIZE THE ACTUAL EXE.
                                0004    186 ;-
                                0004    187
                            00000000    188          .PSECT  $SCB, SHR, EXE, WRT, PAGE
                                0000    189
                                0000    190 SCB_IMAGE::
                    00000000' 0000    191                  .LONG   SCBVECTOR_000   ; UNSPECIFIED. RESERVED TO DEC.
                    00000001' 0004    192 SCB_MCHK:        .LONG   EXE_MCHK+IS     ; MACHINE CHECK ABORT.
                    00000001' 0008    193 SCB_KRNLSTK:     .LONG   EXE_KRNLSTK+IS  ; KERNEL STACK NOT VALID ABORT.
                    00000001' 000C    194 SCB_POWER:       .LONG   EXE_POWER+IS    ; POWER FAIL INTERRUPT.
                    00000000' 0010    195 SCB_OPCDEC:      .LONG   EXE_OPCDEC      ; RESERVED/PRIVILEGED INSTRUCTIONS FAULT.
                    00000000' 0014    196 SCB_OPCCUS:      .LONG   EXE_OPCCUS      ; CUSTOMER RESERVED INSTRUCTION FAULT.
                    00000000' 0018    197 SCB_ROPRAND:     .LONG   EXE_ROPRAND     ; RESERVED OPERAND FAULT/ABORT.
                    00000000' 001C    198 SCB_RADRMOD:     .LONG   EXE_RADRMOD     ; RESERVED ADDRESSING MODE FAULT.
                    00000000' 0020    199 SCB_ACCVIO:      .LONG   EXE_ACCVIO      ; ACCESS CONTROL VIOLATION FAULT.
                    00000000' 0024    200 SCB_TRANSL:      .LONG   EXE_TRANSL      ; TRANSLATION NOT VALID FAULT.
                    00000000' 0028    201 SCB_TBIT:        .LONG   EXE_TBIT        ; TRACE TRAP.
                    00000000' 002C    202 SCB_BREAK:       .LONG   EXE_BREAK       ; BREAKPOINT FAULT.
                    00000000' 0030    203 SCB_COMPAT:      .LONG   EXE_COMPAT      ; COMPATIBILITY TRAP.
                    00000000' 0034    204 SCB_ARITH:       .LONG   EXE_ARITH       ; ARITHMETIC TRAP.
                    00000000' 0038    205                  .LONG   SCBVECTOR_038   ; UNSPECIFIED.
                    00000000' 003C    206                  .LONG   SCBVECTOR_03C   ; UNSPECIFIED.
                    00000000' 0040    207 SCB_CHMK:        .LONG   EXE_CMODKRNL    ; CHMK TRAP.
                    00000000' 0044    208 SCB_CHME:        .LONG   EXE_CHME        ; CHME TRAP.
                    00000000' 0048    209 SCB_CHMS:        .LONG   EXE_CHMS        ; CHMS TRAP.
                    00000000' 004C    210 SCB_CHMU:        .LONG   EXE_CHMU        ; CHMU TRAP.
                    00000000' 0050    211                  .LONG   SCBVECTOR_050   ; UNSPECIFIED.
                    00000000' 0054    212                  .LONG   SCBVECTOR_054   ; UNSPECIFIED.
                    00000000' 0058    213                  .LONG   SCBVECTOR_058   ; UNSPECIFIED.
                    00000000' 005C    214                  .LONG   SCBVECTOR_05C   ; UNSPECIFIED.
                    00000000' 0060    215                  .LONG   SCBVECTOR_060   ; UNSPECIFIED.
                    00000000' 0064    216                  .LONG   SCBVECTOR_064   ; UNSPECIFIED.
                    00000000' 0068    217                  .LONG   SCBVECTOR_068   ; UNSPECIFIED.
                    00000000' 006C    218                  .LONG   SCBVECTOR_06C   ; UNSPECIFIED.
                    00000000' 0070    219                  .LONG   SCBVECTOR_070   ; UNSPECIFIED.
                    00000000' 0074    220                  .LONG   SCBVECTOR_074   ; UNSPECIFIED.
                    00000000' 0078    221                  .LONG   SCBVECTOR_078   ; UNSPECIFIED.
                    00000000' 007C    222                  .LONG   SCBVECTOR_07C   ; UNSPECIFIED.
                    00000000' 0080    223                  .LONG   SCBVECTOR_080   ; UNSPECIFIED.
                    000002A0' 0084    224 SCB_SFTLVL1:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 1 INTERRUPT.
                    000002A0' 0088    225 SCB_SFTLVL2:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 2 INTERRUPT.
                    000002A0' 008C    226 SCB_SFTLVL3:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 3 INTERRUPT.
                    000002A0' 0090    227 SCB_SFTLVL4:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 4 INTERRUPT.
                    000002A0' 0094    228 SCB_SFTLVL5:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 5 INTERRUPT.
                    000002A0' 0098    229 SCB_SFTLVL6:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 6 INTERRUPT.
                    000002A0' 009C    230 SCB_SFTLVL7:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 7 INTERRUPT.
                    000002A0' 00A0    231 SCB_SFTLVL8:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 8 INTERRUPT.
                    000002A0' 00A4    232 SCB_SFTLVL9:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 9 INTERRUPT.
                    000002A0' 00A8    233 SCB_SFTLVL10:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 10 INTERRUPT.
                    000002A0' 00AC    234 SCB_SFTLVL11:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 11 INTERRUPT.
                    000002A0' 00B0    235 SCB_SFTLVL12:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 12 INTERRUPT.
                    000002A0' 00B4    236 SCB_SFTLVL13:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 13 INTERRUPT.
                    000002A0' 00B8    237 SCB_SFTLVL14:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 14 INTERRUPT.
                    000002A0' 00BC    238 SCB_SFTLVL15:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 15 INTERRUPT.
                    00000001' 00C0    239 SCB_TIMER:       .LONG   DSI$TIMSRV+IS   ; INTERVAL TIMER INTERRUPT.
```

```
00000000' 00C4   240          .LONG   SCBVECTOR_0C4   ; UNSPECIFIED.
00000000' 00C8   241          .LONG   SCBVECTOR_0C8   ; UNSPECIFIED.
00000000' 00CC   242          .LONG   SCBVECTOR_0CC   ; UNSPECIFIED.
00000000' 00D0   243          .LONG   SCBVECTOR_0D0   ; UNSPECIFIED.
00000000' 00D4   244          .LONG   SCBVECTOR_0D4   ; UNSPECIFIED.
00000000' 00D8   245          .LONG   SCBVECTOR_0D8   ; UNSPECIFIED.
00000000' 00DC   246          .LONG   SCBVECTOR_0DC   ; UNSPECIFIED.
00000000' 00E0   247          .LONG   SCBVECTOR_0E0   ; UNSPECIFIED.
00000000' 00E4   248          .LONG   SCBVECTOR_0E4   ; UNSPECIFIED.
00000000' 00E8   249          .LONG   SCBVECTOR_0E8   ; UNSPECIFIED.
00000000' 00EC   250          .LONG   SCBVECTOR_0EC   ; UNSPECIFIED.
00000000' 00F0   251          .LONG   SCBVECTOR_0F0   ; UNSPECIFIED.
00000000' 00F4   252          .LONG   SCBVECTOR_0F4   ; UNSPECIFIED.
00000000' 00F8   253          .LONG   SCBVECTOR_0F8   ; UNSPECIFIED.
00000000' 00FC   254          .LONG   SCBVECTOR_0FC   ; UNSPECIFIED.
```

```
                     0100   256          .SBTTL  DSX$INITSCB     INITIALIZE SYSTEM CONTROL BLOCK ROUTINE
                 00000000   257          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                    [23]
                     0000   258
                     0000   259 ;++
                     0000   260 ; FUNCTIONAL DESCRIPTION:
                     0000   261 ;
                     0000   262 ;       This routine generates the necessary SCB.
                     0000   263 ;       The 1st half page contains vectors for exceptions and traps
                     0000   264 ;       The rest of the 4 pages is filled with trap catchers.
                     0000   265 ;       Interrupts are disabled while the SCB is being updated.
                     0000   266 ;       Calls the ONLY_SCB routine to do CPU specific SCB setup.
                     0000   267 ;
                     0000   268 ; CALLING SEQUENCE:
                     0000   269 ;
                     0000   270 ;       DS$INITSCB()
                     0000   271 ;
                     0000   272 ; INPUT PARAMETERS:     NONE
                     0000   273 ;
                     0000   274 ; IMPLICIT INPUTS:      NONE
                     0000   275 ;
                     0000   276 ; OUTPUT PARAMETERS:    NONE
                     0000   277 ;
                     0000   278 ; IMPLICIT OUTPUTS:     NONE
                     0000   279 ;
                     0000   280 ; COMPLETION CODES:     NONE
                     0000   281 ;
                     0000   282 ; SIDE EFFECTS:         NONE
                     0000   283 ;--
```

```
                                 000C  0000    285 .ENTRY  DSX$INITSCB,^M<R2,R3>
                                       0002    286
                                       0002    287         CMK     INITSCB                     ; CHANGE MODE TO KERNEL.
                                       0011    288
                          50    00'    D0  0011    289         MOVL    S^#SS$_NORMAL,R0            ; Set success
                00000000'EF     11    E5  0014    290         BBCC    #DS$V_TIMRON,DS$GL_FLAGS,- ; Clear interval timer in use
                                00        001B    291                 10$
                     00000000'EF      16  001C    292 10$:    JSB     KB_CHECK                    ; CHECK KEYBOARD STATUS.
                                      04  0022    293         RET                                 ; RETURN TO CALLING ROUTINE
                                          0023    294
                                          0023    295 ;
                                          0023    296 ; Reset the whole SCB and reset the SCBB
                                          0023    297 ;
                                          0023    298
                                          C023    299 CMK$INITSCB::
                                   0F     BB  0023    300         PUSHR   #^M<R0,R1,R2,R3>            ; Save registers
                                          0025    301         DSBINT                              ; Disable interrupts
                                   51     D4  002B    302         CLRL    R1                          ; Clear count of quads to be copied
                52  00000000'EF     9E  002D    303         MOVAB   L^SCB_BASE,R2               ; Point to where to build SCB
                53  00000000'EF     9E  0034    304         MOVAB   SCB_IMAGE,R3                ; 1/2 page of SCB image
                                          003B    305
                          82    83    7D  003B    306 10$:    MOVQ    (R3)+,(R2)+                 ; Copy quadword of SCB
                        F9 51     1F    F3  003E    307         AOBLEQ  #31,R1,10$                  ; Copy 1/2 page
                                          0042    308
                53  00000100'EF     9E  0042    309         MOVAB   SCB_UNKINT+^X100,R3         ; Point to unknown interrupt field
                51  000000C4'EF     9E  0049    310         MOVAB   L^UNKNOWN_INT,R1            ; Address of intermediate handler       [23]
                                          0050    311
                     50    03 A3     9E  0050    312 20$:    MOVAB   3(R3),R0                    ; Address after BSBW inst
                   50    51    50     C3  0054    313         SUBL3   R0,R1,R0                    ; make offset to handler
                   50    50    08     78  0058    314         ASHL    #8,R0,R0                    ; Position offset
                         50    30     90  005C    315         MOVB    #^X30,R0                    ; Insert opcode for BSBW
                         83    50     D0  005F    316         MOVL    R0,(R3)+                    ; Insert BSBW UNKNOWN_INT
                      62    FD A3     9E  0062    317         MOVAB   IS-4(R3),(R2)               ; Point SCB vector to BSBW+IS
FFE0 52         04    FFFFFFFC'8F     F1  0066    318         ACBL    #SCB_UNKINT-4,#4,R2,20$     ; For every vector in the SCB
                                          0070    319
                     00000000'EF      16  0070    320         JSB     ONLY_SCB                    ; Do CPU specific SCB setup             [27]
                                          0076    321
                                          0076    322 ;+
                                          0076    323 ; Re-init clock and flag
                                          0076    324 ;-
                                          0076    325
                     00000000'EF      16  0076    326         JSB     L^CLK$RE_INIT               ; Initialize the system timer
                00000000'EF     11    E5  007C    327         BBCC    #DS$V_TIMRON, -             ; RELEASE THE INTERVAL CLOCK.
                                00        0083    328                   DS$GL_FLAGS, 25$
                                          0084    329
                     00000001'EF      D4  0084    330 25$:    CLRL    DS$GA_CHMKVEC               ; CLEAR USER CHMK VECTOR.
                     00000005'EF      D4  008A    331         CLRL    DS$GA_BREAKVEC              ; CLEAR USER BREAK VECTOR.
                     00000009'EF      D4  0090    332         CLRL    DS$GA_TBITVEC               ; CLEAR USER TBIT VECTOR.
                         50    07     D0  0096    333         MOVL    #7,R0                       ; Max software interrupt level
                                          0099    334
                   00000011'EF40     7C  0099    335 30$:    CLRQ    DS$GA_SOFTINT[R0]           ; Clear soft vector
                          F6 50      F5  00A0    336         SOBGTR  R0,30$                      ; Clear all vectors including 0 (mask)
                50  00000000'EF      DE  00A3    337         MOVAL   SCB_BASE,R0                 ; Point to SCB
                        11    50     DA  00AA    338         MTPR    R0,#PR$_SCBB                ; Set SCBB
                00000000'EF     50    D0  00AD    339         MOVL    R0,EXE$GL_SCB               ; Set Virtual address of SCB
                                   02     DD  00B4    340         PUSHL   #DS$_GK_INITSCB             ; Pass fault clear selection code       ;[29
                     00000000'EF      16  00B6    341         JSB     L^DSR$FAULT_CLEAR           ; Reset fault regs and enables
```

```
                     8E    D5  00BC    342          TSTL    (SP)+                       ; Reset the stack
                                00BE    343          ENBINT                          ; Re-enable interrupts
                     0F    BA  00C1    344          POPR    #^M<R0,R1,R2,R3>            ; Restore registers
                           02  00C3    345          REI
                               00C4    346
                               00C4    347 UNKNOWN_INT:
           6E  00000003'8F    C2  00C4    348          SUBL    #SCB_UNKINT+3,(SP)          ; BSBW addr-base is SCB offset
               00000000'EF    16  00CB    349          JSB     DSI$CHANNEL_VEC             ; Enter channel services w/ vector      [23]
               00000000'EF    17  00D1    350          JMP     EXE_UNEXPINT                ; With offset on stack flag unexpected  [23]
```

```
                        352          .SBTTL  DSX$SETVEC      SET VECTOR ROUTINE
                        353 ;++
                        354 ; FUNCTIONAL DESCRIPTION:
                        355 ;
                        356 ;       ROUTINE TO LOAD THE ADDRESS OF AN EXCEPTION OR INTERRUPT SERVICE ROUTINE
                        357 ;       INTO THE SYSTEM CONTROL BLOCK (SCB).
                        358 ;
                        359 ; CALLING SEQUENCE:
                        360 ;
                        361 ;       STANDARD CALL PROCEDURE.
                        362 ;
                        363 ; INPUT PARAMETERS:
                        364 ;
                        365 ;       SETVEC$_VECTOR(AP)      = OFFSET INTO SCB.
                        366 ;       SETVEC$_SRVADR(AP)      = ADDRESS OF INTERRUPT SERVICE ROUTINE
                        367 ;       SETVEC$_CODE(AP)        = VECTOR CODE IN BITS 1:0.
                        368 ;
                        369 ; IMPLICIT INPUTS:      NONE
                        370 ;
                        371 ; OUTPUT PARAMETERS:
                        372 ;
                        373 ;       R0      = COMPLETION CODE.
                        374 ;       R1      = PREVIOUS SERVICE ADDRESS.
                        375 ;
                        376 ; IMPLICIT OUTPUTS:      NONE
                        377 ;
                        378 ; COMPLETION CODES:
                        379 ;
                        380 ;       DS$_NORMAL      = VECTOR MODIFIED.
                        381 ;       DS$_IVADDR      = SERVICE ADDRESS <1:0> NOT ZERO.
                        382 ;       DS$_IVVECT      = INVALID VECTOR NUMBER.
                        383 ;       DS$_ICBUSY      = INTERVAL CLOCK BUSY.
                        384 ;
                        385 ; SIDE EFFECTS:         NONE
                        386 ;--
```

```
                        000C  00D7   388 .ENTRY  DSX$SETVEC,^M<R2,R3>          ; Save R2 and R3                          [24]
                              00D9   389
                 013F   30    00D9   390         BSBW    CHECK_VECTOR          ; Check vector for validity
                              00DC   391
     50  00660040 8F   D0    00DC   392         MOVL    #DS$_IVADDR, R0       ; SET ERROR RETURN CODE.
          08 AC    03   D3    00E3   393         BITL    #3,8(AP)              ; CHECK FOR LONGWORD ALIGNED ADR.
                   49   12    00E7   394         BNEQ    53$                   ; Exit if not longword aligned
                              00E9   395
     51  00000000'E2   DE    00E9   396         MOVAL   L^SCB_BASE(R2), R1    ; GET ADR OF VECTOR AS I KNOW IT.
                              00F0   397
          0040 8F   52   B1    00F0   398         CMPW    R2, #SCB$L_CHMK       ; CHECK FOR CHMK VECTOR.
                   09   12    00F5   399         BNEQ    30$
     51  00000001'EF   DE    00F7   400         MOVAL   DS$GA_CHMKVEC, R1     ; SET SOFT VECTOR INSTEAD.
                   6A   11    00FE   401         BRB     100$
                              0100   402
               2C  52   B1    0100   403 30$:    CMPW    R2, #SCB$L_BREAK      ; CHECK FOR BREAK VECTOR.
                   09   12    0103   404         BNEQ    40$
     51  00000005'EF   DE    0105   405         MOVAL   DS$GA_BREAKVEC, R1    ; SET SOFT VECTOR INSTEAD.
                   5C   11    010C   406         BRB     100$
                              010E   407
               28  52   B1    010E   408 40$:    CMPW    R2, #SCB$L_TBIT       ; CHECK FOR TBIT VECTOR.
                   09   12    0111   409         BNEQ    50$
     51  00000009'EF   DE    0113   410         MOVAL   DS$GA_TBITVEC, R1     ; SET SOFT VECTOR INSTEAD.
                   4E   11    011A   411         BRB     100$
                              011C   412
          00C0 8F   52   B1    011C   413 50$:    CMPW    R2, #SCB$L_TIMER      ; CHECK FOR TIMER VECTOR.
                   2A   12    0121   414         BNEQ    60$
     00000000'EF   11   E3    0123   415         BBCS    #DS$V_TIMRON, -       ; SET TIMER FLAG.
                   09         012A   416                   DS$GL_FLAGS, 55$
     50  006600C8 8F   D0    012B   417         MOVL    #DS$_ICBUSY, R0       ; INTERVAL CLOCK BUSY.
                              0132   418
                   57   11    0132   419 53$:    BRB     DSX$SETVEC_X
                              0134   420
     00000000'EF   01   90    0134   421 55$:    MovB    #1, L^DS$GB_Stopping_The_Timer  ; Set flag to signal stopping   [24]
                              013B   422
                              013B   423         ;+
                              013B   424         ; Do an Mtpr and then an Mfpr because the interval clock may interrupt in
                              013B   425         ; between the time the MTPR console code is executing and when the MTPR
                              013B   426         ; console code is actually finished. The Mtpr console code may
                              013B   427         ; actually be executing at the same time the MTPR macro code starts.
                              013B   428         ;-
                              013B   429
     18  80000080 8F   DA    013B   430         MTPR    #ICCS$M_ERR + -       ; CLEAR ERROR &
                              0142   431                   ICCS$M_INT , -      ; INTERRUPT & STOP THE CLOCK
                              0142   432                   #PR$_ICCS           ; IN THE INTERVAL TIMER
                              0142   433
               53  18   DB    0142   434         Mfpr    #PR$_ICCS, R3         ; This ensures that the preceding         [25]
                              0145   435                                       ; MTPR is really finished (at the         [25]
                              0145   436                                       ; console level)                          [25]
                              0145   437
          00000000'EF   94    0145   438         ClrB    L^DS$GB_Stopping_The_Timer    ; Clear flag to signal done     [24]
                   1D   11    014B   439         BRB     100$
                              014D   440
          0084 8F   52   B1    014D   441 60$:    CMPW    R2,#SCB$L_SFTLVL1     ; Possible software interrupt vector
                   16   19    0152   442         BLSS    70$                   ; No, desired vector is too low
          00BC 8F   52   B1    0154   443         CMPW    R2,#SCB$L_SFTLVL15    ; Possible software interrupt vector
                   0F   14    0159   444         BGTR    70$                   ; No, desired vector is too high
```

```
                51   52   04   02   EF  015B   445          EXTZV   #2,#4,R2,R1              ; Isolate software interrupt level
              51   00000011'EF41   DE  0160   446          MOVAL   DS$GA_SOFTINT[R1],R1      ; Point to true vector
                           00   11  0168   447          BRB     100$
                                    016A   448
                                    016A   449 70$:
                                    016A   450
                                    016A   451 ;+
                                    016A   452 ; Vector offset in R2
                                    016A   453 ; Address of vector in R1
                                    016A   454 ;-
                                    016A   455
                           61   DD  016A   456 100$:   PUSHL   (R1)                        ; SAVE OLD SER'.      (ESS.
                      08 AC   D5  016C   457          TSTL    8(AP)                       ; See if address is                    [21]
                         04   13  016F   458          BEQL    110$                        ; If so, don't change                  [21]
                    61   08 AC   D0  0171   459          MOVL    8(AP), (R1)                 ; LOAD THE VECTOR REQUESTED
                                    0175   460
                                    0175   461 110$:                                      ;                                      [21]
                51   00000000'E2   DE  0175   462          MOVAL   L^SCB_BASE(R2), R1          ; Set back to SCB vec (soft vector)    [21]
            61   02   00   0C AC   F0  017C   463          INSV    12(AP), #0, #2, (R1)        ; SET VECTOR CODE BITS 1:0.
                         02   BA  0182   464          POPR    #^M<R1>                     ; PUT OLD SERVICE ADDRESS INTO R1.
                50   00660001 8F   D0  0184   465          MOVL    #DS$_NORMAL, R0             ; SET SUCCESSFUL RETURN
                                    018B   466
                                    018B   467 DSX$SETVEC_X:
                    00000000'EF   16  018B   468          JSB     KB_CHECK                    ; CHECK KEYBOARD STATUS.
                                  04  0191   469          RET                                 ; RETURN TO THE DIAGNOSTIC
```

```
                                .SBTTL  DSX$CLRVEC      RESTORE VECTOR ROUTINE
                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ; THIS ROUTINE LOADS THE VECTOR IN THE 'SCB' WITH THE CONTENTS
                        ; OF THE CORRESPONDING VECTOR IN 'SCB_IMAGE' (A PAGE WHICH HOLDS THE
                        ; INITIAL CONTENTS OF EACH VECTOR).
                        ; Interrupts are disabled while the SCB is being changed.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       DS$CLRVEC(VECTOR OFFSET)
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       4(AP) OFFSET INTO S.C.B.
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       SCB_IMAGE - INITIAL SYSTEM CONTROL BLOCK
                        ;
                        ; OUTPUT PARAMETERS:
                        ;
                        ;       R0 = 1 > SUCCESSFUL
                        ;       R0 = 0 > FAILURE
                        ;
                        ; IMPLICIT OUTPUTS:      NONE
                        ;
                        ; COMPLETION CODES:      NONE
                        ;
                        ; SIDE EFFECTS:          NONE
                        ;--
```

```
                        0004  0192   504 .ENTRY  DSX$CLRVEC,^M<R2>
                              0194   505
                        0084   30  0194   506          BSBW    CHECK_VECTOR            ; Verify vector
                              0197   507
              0040 8F    52   B1  0197   508 10$:     CMPW    R2, #SCB$L_CHMK         ; CHECK FOR CHMK VECTOR.
                         08   12  019C   509          BNEQ    20$
               000C0001'EF   D4  019E   510          CLRL    DS$GA_CHMKVEC           ; CLEAR SOFT VECTOR.
                         36   11  01A4   511          BRB     100$
                              01A6   512
                     2C    52   B1  01A6   513 20$:     CMPW    R2, #SCB$L_BREAK        ; CHECK FOR BREAK VECTOR.
                         08   12  01A9   514          BNEQ    30$
               00000005'EF   D4  01AB   515          CLRL    DS$GA_BREAKVEC          ; CLEAR SOFT VECTOR.
                         29   11  01B1   516          BRB     100$
                              01B3   517
                     28    52   B1  01B3   518 30$:     CMPW    R2, #SCB$L_TBIT         ; CHECK FOR TBIT VECTOR.
                         08   12  01B6   519          BNEQ    50$
               00000009'EF   D4  01B8   520          CLRL    DS$GA_TBITVEC           ; CLEAR SOFT VECTOR.
                         1C   11  01BE   521          BRB     100$
                              01C0   522
              0084 8F    52   B1  01C0   523 50$:     CMPW    R2,#SCB$L_SFTLVL1       ; Possible software interrupt vector
                         15   19  01C5   524          BLSS    60$                     ; No, desired vector is too low
              00BC 8F    52   B1  01C7   525          CMPW    R2,#SCB$L_SFTLVL15      ; Possible software interrupt vector
                         0E   14  01CC   526          BGTR    60$                     ; No, desired vector is too high
          50   52   04   02   EF  01CE   527          EXTZV   #2,#4,R2,R0             ; Isolate software interrupt level
               00000011'EF40   D4  01D3   528          CLRL    DS$GA_SOFTINT[R0]       ; Clear vector
                         00   11  01DA   529          BRB     100$
                              01DC   530
                              01DC   531 60$:
     50   00000001'E2   9E  01DC   532 100$:    MOVAB   L^SCB_UNKINT+IS(R2),R0  ; Restore unexpected handler
          0100 8F    52   B1  01E3   533          CMPW    R2,#^X100               ; 1st 1/2 page?
                         07   18  01E8   534          BGEQ    120$                    ; Branch if not
     50   00000000'E2   D0  01EA   535          MOVL    L^SCB_IMAGE(R2),R0      ; Get static vector
                              01F1   536
     00000000'E2   50   D0  01F1   537 120$:    MOVL    R0,L^SCB_BASE(R2)       ; RESTORE THE VECTOR.
          00C0 8F    52   B1  01F8   538          CMPW    R2, #SCB$L_TIMER        ; TIMER VECTOR?                         [28]
                         0E   12  01FD   539          BNEQ    150$                    ; No                                    [28]
               00000000'EF   16  01FF   540          JSB     L^CLK$RE_INIT           ; Yes, initialize the system timer      [28]
     00000000'EF   11   E5  0205   541          BBCC    #DS$V_TIMRON, -         ; RELEASE THE INTERVAL CLOCK.           [28]
                         00   0209   542            DS$GL_FLAGS, 150$     ;                                       [28]
     50   00000000'8F   D0  020D   543 150$:    MOVL    #SS$_NORMAL, R0         ; SET SUCCESSFUL RETURN
                              0214   544
                              0214   545 DSX$CLRVEC_X:
               00000000'EF   16  0214   546          JSB     KB_CHECK                ; CHECK KEYBOARD.
                         04   021A   547          RET                             ; RETURN TO DIAGNOSTIC PROGRAM
```

```
                     021B    549          .SBTTL  CHECK_VECTOR    Verify vector range
                     021B    550 ;++
                     021B    551 ; FUNCTIONAL DESCRIPTION:
                     021B    552 ;
                     021B    553 ;       This routine is used to verify that the vector specified
                     021B    554 ;       falls within the SCB and is aligned properly
                     021B    555 ;
                     021B    556 ; CALLING SEQUENCE:
                     021B    557 ;
                     021B    558 ;       BSBW    CHECK_VECTOR
                     021B    559 ;
                     021B    560 ; INPUT PARAMETERS:
                     021B    561 ;
                     021B    562 ;       4(AP)   Vector
                     021B    563 ;
                     021B    564 ; IMPLICIT INPUTS:      NONE
                     021B    565 ;
                     021B    566 ; IMPLICIT INPUTS:      NONE
                     021B    567 ;
                     021B    568 ; OUTPUT PARAMETERS:    NONE
                     021B    569 ;
                     021B    570 ; IMPLICIT OUTPUTS:
                     021B    571 ;
                     021B    572 ;       R2      correct vector
                     021B    573 ;
                     021B    574 ; SIDE EFFECTS:
                     021B    575 ;
                     021B    576 ;       Returns to caller if vector bad
                     021B    577 ;
                     021B    578 ; COMPLETION CODES:
                     021B    579 ;
                     021B    580 ;--
```

```
                                  021B    582 CHECK_VECTOR:
          52    04 AC    D0       021B    583          MOVL    4(AP),R2                    ; Get vector
                   0F    13       021F    584          BEQL    10$                         ; Branch if bad
     52   00000000'8F    D1       0221    585          CMPL    #SCB_UNKINT-SCB_BASE,R2     ; Within SCB?
                   06    15       0228    586          BLEQ    10$                         ; Branch if not
             52    03    D3       022A    587          BITL    #3,R2                       ; Longword aligned?
                   01    12       022D    588          BNEQ    10$                         ; Branch if not
                         05       022F    589          RSB                                 ; Return since vector ok
                                  0230    590
     50   00660038 8F    D0       0230    591 10$:     MOVL    #DS$_IVVECT,R0              ; Sorry bad vector, return
                         04       0237    592          RET
```

```
0238   594          .SBTTL  DSX$SETIPL      SET INTERRUPT PRIORITY LEVEL SERVICE PROCEDURE.
0238   595 ;++
0238   596 ; FUNCTIONAL DESCRIPTION:
0238   597 ;
0238   598 ;       THIS ROUTINE SETS THE CURRENT IPL
0238   599 ;
0238   600 ; CALLING SEQUENCE:     NONE
0238   601 ;
0238   602 ; INPUT PARAMETERS:     NONE
0238   603 ;
0238   604 ; IMPLICIT INPUTS:      NONE
0238   605 ;
0238   606 ; OUTPUT PARAMETERS:    NONE
0238   607 ;
0238   608 ; IMPLICIT OUTPUTS:     NONE
0238   609 ;
0238   610 ; COMPLETION CODES:     NONE
0238   611 ;
0238   612 ; SIDE EFFECTS:         NONE
0238   613 ;--
```

```
                                0000  0238   615 .ENTRY  DSX$SETIPL,^M<>
                                      023A   616
                                      023A   617          CMK     SETIPL                          ; Change mode to kernel
                50   00000000'8F     D0   0249   618          MOVL    #SS$_NORMAL,R0                  ; INDICATE SUCCESS
                                      04   0250   619          RET
                                           0251   620
                                           0251   621 CMK$SETIPL::
                     12   04 AC     DA   0251   622          MTPR    4(AP), #PR$_IPL
          04 AE      05   10   04 AC   F0   0255   623          INSV    4(AP),#PSL$V_IPL, -
                                           025C   624                  #PSL$S_IPL,4(SP)        ; Set IPL also in PSL about to be.
                                      02   025C   625          REI                             ; Exit CHMK
```

```
                        025D   627          .SBTTL  SOFT$INT        Software interrupt vectors
                        025D   628 ;++
                        025D   629 ; FUNCTIONAL DESCRIPTION:
                        025D   630 ;
                        025D   631 ;        These vectors are the software interrupt vector.
                        025D   632 ;
                        025D   633 ;--
                        025D   634          .ALIGN  LONG
                        0260   635 SOFT$INT_A:
00000000'               0260   636          .ADDRESS        0               ; IPL 0 impossible / filler
00000000'               0264   637          .ADDRESS        0               ; IPL 1 usused
00000000'               0268   638          .ADDRESS        SCH$ASTDEL      ; IPL 2 is AST delivery
00000000'               026C   639          .ADDRESS        0               ; IPL 3 unused
00000000'               0270   640          .ADDRESS        IOC$IOPOST      ; IPL 4 QIO post processing
00000000'               0274   641          .ADDRESS        DS$SOFTIPL5     ; IPL 5 Enter special handler
00000000'               0278   642          .ADDRESS        EXE$FORKDSPTH   ; IPL 6 Fork dispatch
00000000'               027C   643          .ADDRESS        DSI$TIMER       ; IPL 7 Timer interrupt service
00000000'               0280   644          .ADDRESS        EXE$FORKDSPTH   ; IPL 8 Fork dispatch
00000000'               0284   645          .ADDRESS        EXE$FORKDSPTH   ; IPL 9 Fork dispatch
00000000'               0288   646          .ADDRESS        EXE$FORKDSPTH   ; IPL 10 Fork dispatch
00000000'               028C   647          .ADDRESS        EXE$FORKDSPTH   ; IPL 11 Fork dispatch
00000000'               0290   648          .ADDRESS        0               ; IPL 12 unused
00000000'               0294   649          .ADDRESS        0               ; IPL 13 unused
00000000'               0298   650          .ADDRESS        0               ; IPL 14 unused
00000000'               029C   651          .ADDRESS        0               ; IPL 15 unused
                        02A0   652
                        02A0   653
                        02A0   654          .ALIGN  LONG                    ; Software interrupt vector (all IPL's)
                        02A0   655 SOFT$INT:
                  03    BB     02A0   656          PUSHR   #^M<R0,R1>      ; Save register
                50  12    DB     02A2   657          MFPR    #PR$_IPL,R0     ; Get IPL
0A 00000011'EF    50    E4     02A5   658          BBSC    R0,DS$GA_SOFTINT,10$    ; Branch if supervisor expected this
 51    00000011'EF  40    D0     02AD   659          MOVL    DS$GA_SOFTINT[R0],R1    ; User handler address?
                  1B    12     02B5   660          BNEQ    20$             ; Branch if user handler
                        02B7   661
      51    A5 AF  40    D0     02B7   662 10$:     MOVL    SOFT$INT_A[R0],R1       ; Get handler address
                  25    12     02BC   663          BNEQ    30$             ; Branch if handler specified
        51    04 AE    D0     02BE   664          MOVL    4(SP),R1        ; Restore R1
            50  20    C0     02C2   665          ADDL    #^X20,R0        ; Generate SCB offset
   04 AE    50  02    78     02C5   666          ASHL    #2,R0,4(SP)     ; SCB vector of unexpected interrupt
                  01    BA     02CA   667          POPR    #^M<R0>         ; Restore register
     00000000'EF    17    02CC   668          JMP     EXE_UNEXPINT            ; Process unexpected interrupt          [23]
                        02D2   669
        7E    6E    7D     02D2   670 20$:     MOVQ    (SP),-(SP)      ; Make space for PC/PSL
      08 AE    51    D0     02D5   671          MOVL    R1,8(SP)        ; Set PC
            0C AE    DC     02D9   672          MOVPSL  12(SP)          ; Current PSL for interrupt routine
      51    80 AF  40    D0     02DC   673          MOVL    SOFT$INT_A[R0],R1       ; Get supervisor handler
                  0A    13     02E1   674          BEQL    40$             ; Branch if none
                        02E3   675
        7E    6E    7D     02E3   676 30$:     MOVQ    (SP),-(SP)      ; make space for PC/PSL
      08 AE    51    D0     02E6   677          MOVL    R1,8(SP)        ; Set PC
            0C AE    DC     02EA   678          MOVPSL  12(SP)          ; Current PSL for interrupt routine
                        02ED   679
                  03    BA     02ED   680 40$:     POPR    #^M<R0,R1>      ; Restore
                        02    02EF   681          REI                     ; Enter interrupt handler
                        02F0   682
                        02F0   683          .END
```

```
$$ARGS                          =  00000003     D
$$T1                            =  00000010     D
$ER                             =  00000001     D
$MODULE                            00000000 R   D   03
BIT...                          =  00000004     D
CHECK_VECTOR                       0000021B R   D   05
CLK$RE_INIT                        ********   X     05
CMK$INITSCB                        00000023 RG  D   05
CMK$SETIPL                         00000251 RG  D   05
CMK$_                           =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
CVTREG$_DATA                    =  00000008     D
CVTREG$_MAXLEN                  =  00000014     D
CVTREG$_MNEADR                  =  0000000C     D
CVTREG$_MSB                     =  00000004     D
CVTREG$_NARGS                   =  0000000B     D
CVTREG$_STRBUF                  =  00000010     D
CVTREG$_V1                      =  00000018     D
CVTREG$_V2                      =  0000001C     D
CVTREG$_V3                      =  00000020     D
CVTREG$_V4                      =  00000024     D
CVTREG$_V5                      =  00000028     D
CVTREG$_V6                      =  0000002C     D
DS$GA_BREAKVEC                     00000005 RG  D   02
DS$GA_CHMKVEC                      00000001 RG  D   02
DS$GA_SOFTINT                      00000011 RG  D   02
DS$GA_TBITVEC                      00000009 RG  D   02
DS$GB_STOPPING_THE_TIMER           00000000 RG  D   02
DS$GL_FLAGS                        ********     X     05
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_WARNING                    =  00000000     D
DS$M_ABRTFLG                    =  00000040     D
DS$M_BADTIME                    =  00100000     D
DS$M_BATCH                      =  00400000     D
DS$M_BRKCLR                     =  00001000     D
DS$M_BRKPT                      =  00000800     D
DS$M_CHARFLG                    =  00000100     D
DS$M_CMDFLG                     =  00000080     D
DS$M_CTRLC                      =  00000001     D
DS$M_CTRLO                      =  00010000     D
DS$M_DEVFLG                     =  00000200     D
DS$M_DISABLCC                   =  01000000     D
DS$M_DONFLG                     =  00002000     D
DS$M_ERRFLG                     =  00000010     D
DS$M_EXCEPT                     =  00080000     D
DS$M_EXETST                     =  00040000     D
DS$M_HLTFLG                     =  00000008     D
DS$M_LODFLG                     =  00000002     D
DS$M_MEMMGT                     =  00008000     D
DS$M_OUTPUT                     =  00800000     D
DS$M_RUBFLG                     =  00000020     D
DS$M_SCRIPT                     =  00200000     D
DS$M_SETIMR                     =  02000000     D
DS$M_STRFLG                     =  00000004     D
DS$M_SUBT                       =  00004000     D
DS$M_SYSFLG                     =  00000400     D
DS$M_TIMRON                     =  00020000     D
DS$SOFTIPL5                        ********     X     05
DS$V_ABRTFLG                    =  00000006     D
DS$V_BADTIME                    =  00000014     D
DS$V_BATCH                      =  00000016     D
DS$V_BRKCLR                     =  0000000C     D
DS$V_BRKPT                      =  0000000B     D
DS$V_CHARFLG                    =  00000008     D
DS$V_CMDFLG                     =  00000007     D
DS$V_CTRLC                      =  00000000     D
DS$V_CTRLO                      =  00000010     D
DS$V_DEVFLG                     =  00000009     D
DS$V_DISABLCC                   =  00000018     D
DS$V_DONFLG                     =  0000000D     D
DS$V_ERRFLG                     =  00000004     D
DS$V_EXCEPT                     =  00000013     D
DS$V_EXETST                     =  00000012     D
DS$V_HLTFLG                     =  00000003     D
DS$V_LODFLG                     =  00000001     D
DS$V_MEMMGT                     =  0000000F     D
DS$V_OUTPUT                     =  00000017     D
DS$V_RUBFLG                     =  00000005     D
DS$V_SCRIPT                     =  00000015     D
DS$V_SETIMR                     =  00000019     D
DS$V_STRFLG                     =  00000002     D
DS$V_SUBT                       =  0000000E     D
DS$V_SYSFLG                     =  0000000A     D
DS$V_TIMRON                     =  00000011     D
DS$_ARITH                       =  006600D0     D
DS$_ASBE                        =  00660118     D
DS$_BADLINK                     =  006600F0     D
DS$_BADTYPE                     =  006600E8     D
DS$_BIIC                        =  00660120     D
DS$_CHME                        =  006600A8     D
DS$_CHMK                        =  006600E0     D
DS$_DEVNAME                     =  00660108     D
DS$_ERROR                       =  00660002     D
DS$_HWE                         =  00660068     D
DS$_FRAGBUF                     =  00660080     D
DS$_GK_EXE_MCHK                 =  00000000     D
DS$_GK_INITSCB                  =  00000002     D
DS$_GK_INT_WORK                 =  00000003     D
```

```
DS$_GK_TST$MCHK                 =  00000001     D
DS$_ICBUSY                      =  006600C8     D
DS$_ICERR                       =  006600C0     D
DS$_IHWE                        =  00660060     D
DS$_ILLCHAR                     =  00660018     D
DS$_ILLPAGCNT                   =  00660078     D
DS$_ILLUNIT                     =  00660100     D
DS$_INSFMEM                     =  00660050     D
DS$_IPL2HI                      =  006600B8     D
DS$_IVADDR                      =  00660040     D
DS$_IVVECT                      =  00660038     D
DS$_KRNLSTK                     =  00660090     D
DS$_LOGIC                       =  00660070     D
DS$_MCHK                        =  00660088     D
DS$_MMOFF                       =  00660058     D
DS$_NEEDUNIT                    =  006600F8     D
DS$_NODE                        =  00660128     D
DS$_NOPCS                       =  00660110     D
DS$_NORMAL                      =  00660001     D
DS$_NOSUPPORT                   =  006600B1     D
DS$_NOTDON                      =  00660030     D
DS$_NOTIMP                      =  006600B0     D
DS$_NULLSTR                     =  00660010     D
DS$_OVERFLOW                    =  00660008     D
DS$_POWER                       =  00660098     D
DS$_PROGERR                     =  00660020     D
DS$_SEVERE                      =  00660004     D
DS$_TRANSL                      =  006600A0     D
DS$_TRUNCATE                    =  00660028     D
DS$_UNEXPINT                    =  006600D8     D
DS$_VASFULL                     =  00660048     D
DS$_WARNING                     =  00660000     D
DSI$CHANNEL_VEC                    ********     X    05
DSI$TIMER                          ********     X    05
DSI$TIMSRV                         ********     X    04
DSR$FAULT_CLEAR                    ********     X    05
DSX$CLRVEC                         00000192 RG  D    05
DSX$CLRVEC_X                       00000214 R   D    05
DSX$INITSCB                        00000000 RG  D    05
DSX$SETIPL                         00000238 RG  D    05
DSX$SETVEC                         000000D7 RG  D    05
DSX$SETVEC_X                       00000188 R   D    05
EXE$FORKDSPTH                      ********     X    05
EXE$GL_SCB                         0000000D RG  D    02
EXE_ACCVIO                         ********     X    04
EXE_ARITH                          ********     X    04
EXE_BREAK                          ********     X    04
EXE_CHME                           ********     X    04
EXE_CHMS                           ********     X    04
EXE_CHMU                           ********     X    04
EXE_CMODKRNL                       ********     X    04
EXE_COMPAT                         ********     X    04
EXE_KRNLSTK                        ********     X    04
EXE_MCHK                           ********     X    04
EXE_OPCCUS                         ********     X    04
EXE_OPCDEC                         ********     X    04
EXE_POWER                          ********     X    04
EXE_RADRMOD                        ********     X    04
EXE_ROPRAND                        ********     X    04
EXE_TBIT                           ********     X    04
EXE_TRANSL                         ********     X    04
EXE_UNEXPINT                       ********     X    05
ICCS$M_ERR                      =  80000000     D
ICCS$M_INT                      =  00000080     D
IOC$IOPOST                         ********     X    05
IS                              =  00000001     D
KB_CHECK                           ********     X    05
ONLY_SCB                           ********     X    05
PR$_ICCS                        =  00000018     D
PR$_IPL                         =  00000012     D
PR$_SCBB                        =  00000011     D
PSL$S_IPL                       =  00000005     D
PSL$V_IPL                       =  00000010     D
PSL$V_IS                        =  0000001A     D
SCB$L_ACCESS                       00000020     D
SCB$L_ARITH                        00000034     D
SCB$L_BREAK                        0000002C     D
SCB$L_CHME                         00000044     D
SCB$L_CHMK                         00000040     D
SCB$L_CHMS                         00000048     D
SCB$L_CHMU                         0000004C     D
SCB$L_COMPAT                       00000030     D
SCB$L_KNLSTK                       00000008     D
SCB$L_MACHCHK                      00000004     D
SCB$L_OPCCUS                       00000014     D
SCB$L_OPCDEC                       00000010     D
SCB$L_POWER                        0000000C     D
SCB$L_RADRMOD                      0000001C     D
SCB$L_ROPRAND                      00000018     D
SCB$L_RXDB                         000000F8     D
SCB$L_SFTLVL1                      00000084     D
SCB$L_SFTLVL10                     000000A8     D
SCB$L_SFTLVL11                     000000AC     D
SCB$L_SFTLVL12                     000000B0     D
SCB$L_SFTLVL13                     000000B4     D
SCB$L_SFTLVL14                     000000B8     D
SCB$L_SFTLVL15                     000000BC     D
SCB$L_SFTLVL2                      00000088     D
SCB$L_SFTLVL3                      0000008C     D
SCB$L_SFTLVL4                      00000090     D
SCB$L_SFTLVL5                      00000094     D
SCB$L_SFTLVL6                      00000098     D
SCB$L_SFTLVL7                      0000009C     D
SCB$L_SFTLVL8                      000000A0     D
SCB$L_SFTLVL9                      000000A4     D
SCB$L_TBIT                         00000028     D
SCB$L_TIMER                        000000C0     D
SCB$L_TRANSL                       00000024     D
SCB$L_TXDB                         000000FC     D
SCB$L_ZERO                         00000000     D
SCBVECTOR_000                      ********     X    04
SCBVECTOR_038                      ********     X    04
SCBVECTOR_03C                      ********     X    04
SCBVECTOR_050                      ********     X    04
```

```
SCBVECTOR_054                   ********    X   04
SCBVECTOR_058                   ********    X   04
SCBVECTOR_05C                   ********    X   04
SCBVECTOR_060                   ********    X   04
SCBVECTOR_064                   ********    X   04
SCBVECTOR_068                   ********    X   04
SCBVECTOR_06C                   ********    X   04
SCBVECTOR_070                   ********    X   04
SCBVECTOR_074                   ********    X   04
SCBVECTOR_078                   ********    X   04
SCBVECTOR_07C                   ********    X   04
SCBVECTOR_080                   ********    X   04
SCBVECTOR_0C4                   ********    X   04
SCBVECTOR_0C8                   ********    X   04
SCBVECTOR_0CC                   ********    X   04
SCBVECTOR_0D0                   ********    X   04
SCBVECTOR_0D4                   ********    X   04
SCBVECTOR_0D8                   ********    X   04
SCBVECTOR_0DC                   ********    X   04
SCBVECTOR_0E0                   ********    X   04
SCBVECTOR_0E4                   ********    X   04
SCBVECTOR_0E8                   ********    X   04
SCBVECTOR_0EC                   ********    X   04
SCBVECTOR_0F0                   ********    X   04
SCBVECTOR_0F4                   ********    X   04
SCBVECTOR_0F8                   ********    X   04
SCBVECTOR_0FC                   ********    X   04
SCB_ACCVIO                      00000020 R  D   04
SCB_ARITH                       00000034 R  D   04
SCB_BASE                        ********    X   05
SCB_BREAK                       0000002C R  D   04
SCB_CHME                        00000044 R  D   04
SCB_CHMK                        00000040 R  D   04
SCB_CHMS                        00000048 R  D   04
SCB_CHMU                        0000004C R  D   04
SCB_COMPAT                      00000030 R  D   04
SCB_IMAGE                       00000000 RG D   04
SCB_KRNLSTK                     00000008 R  D   04
SCB_MCHK                        00000004 R  D   04
SCB_OPCCUS                      00000014 R  D   04
SCB_OPCDEC                      00000010 R  D   04
SCB_POWER                       0000000C R  D   04
SCB_RADRMOD                     0000001C R  D   04
SCB_ROPRAND                     00000018 R  D   04
SCB_SFTLVL1                     00000084 R  D   04
SCB_SFTLVL10                    000000A8 R  D   04
SCB_SFTLVL11                    000000AC R  D   04
SCB_SFTLVL12                    000000B0 R  D   04
SCB_SFTLVL13                    000000B4 R  D   04
SCB_SFTLVL14                    000000B8 R  D   04
SCB_SFTLVL15                    000000BC R  D   04
SCB_SFTLVL2                     00000088 R  D   04
SCB_SFTLVL3                     0000008C R  D   04
SCB_SFTLVL4                     00000090 R  D   04
SCB_SFTLVL5                     00000094 R  D   04
SCB_SFTLVL6                     00000098 R  D   04
SCB_SFTLVL7                     0000009C R  D   04
SCB_SFTLVL8                     000000A0 R  D   04
SCB_SFTLVL9                     000000A4 R  D   04
SCB_TBIT                        00000028 R  D   04
SCB_TIMER                       000000C0 R  D   04
SCB_TRANSL                      00000024 R  D   04
SCB_UNKINT                      ********    X   05
SCH$ASTDEL                      ********    X   05
SETVEC$_CODE                =   0000000C    D
SETVEC$_NARGS               =   00000003    D
SETVEC$_SRVADR              =   00000008    D
SETVEC$_VECTOR              =   00000004    D
SIZ...                      =   00000001    D
SOFT$INT                        000002A0 R  D   05
SOFT$INT_A                      00000260 R  D   05
SS$_NORMAL                      ********    X   05
UNKNOWN_INT                     000000C4 R  D   05
```

SYSTEM CONTROL BLOCK

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000200 (512.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000051 (81.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000004 (4.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
$SCB	00000100 (256.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC PAGE
CODE	000002F0 (752.)	05 (5.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000003	145 (1)	140 (1) 145 (1)
$$T1	=00000010	145 (1)	140 (1) 145 (1)
$ER	=00000001	181 (1)	
$MODULE	00000000-R	181 (1)	
BIT...	=00000004	148 (1)	141 (1) 146 (1) 147 (1) 148 (1)
CHECK_VECTOR	0000021B-R	582 (1)	#-390 (1) #-506 (1)
CLK$RE_INIT	00000000-XR		326 (1) 540 (1)
CMK$INITSCB	00000023-R	299 (1)	#-287 (1)
CMK$SETIPL	00000251-R	621 (1)	#-617 (1)
CMK$_	=00000004	146 (1)	
CMK$_ASTEXIT	=00000000	146 (1)	
CMK$_CANTIM	=00000008	146 (1)	
CMK$_HIBER	=00000007	146 (1)	
CMK$_INITSCB	=00000008	146 (1)	#-287 (1)
CMK$_LAST	=0000000E	146 (1)	
CMK$_MMENABLE	=00000003	146 (1)	
CMK$_RCONSOLE	=00000001	146 (1)	
CMK$_REFRESH	=00000005	146 (1)	
CMK$_SCHDWK	=00000006	146 (1)	
CMK$_SETIMR	=0000000C	146 (1)	
CMK$_SETIPL	=00000009	146 (1)	#-617 (1)
CMK$_SETPRT	=0000000D	146 (1)	
CMK$_SGIPR	=0000000A	146 (1)	
CMK$_TCONSOLE	=00000002	146 (1)	
CVTREG$_DATA	=00000008	140 (1)	
CVTREG$_MAXLEN	=00000014	140 (1)	
CVTREG$_MNEADR	=0000000C	140 (1)	
CVTREG$_MSB	=00000004	140 (1)	
CVTREG$_NARGS	=0000000B	140 (1)	
CVTREG$_STRBUF	=00000010	140 (1)	
CVTREG$_V1	=00000018	140 (1)	
CVTREG$_V2	=0000001C	140 (1)	
CVTREG$_V3	=00000020	140 (1)	
CVTREG$_V4	=00000024	140 (1)	
CVTREG$_V5	=00000028	140 (1)	
CVTREG$_V6	=0000002C	140 (1)	
DS$GA_BREAKVEC	00000005-R	166 (1)	#-331 (1) 405 (1) #-515 (1)
DS$GA_CHMKVEC	00000001-R	163 (1)	#-330 (1) 400 (1) #-510 (1)
DS$GA_SOFTINT	00000011-R	175 (1)	#-335 (1) 446 (1) #-528 (1) 658 (1)
			#-659 (1)
DS$GA_TBITVEC	00000009-R	169 (1)	#-332 (1) 410 (1) #-520 (1)
DS$GB_STOPPING_THE_TIMER	00000000-R	160 (1)	#-421 (1) #-438 (1)
DS$GL_FLAGS	00000000-XR		290 (1) 328 (1) 416 (1) 542 (1)
DS$K_ERROR	=00000002	141 (1)	
DS$K_NORMAL	=00000001	141 (1)	
DS$K_SEVERE	=00000004	141 (1)	
DS$K_SUBSYS	=00000066	141 (1)	141 (1)
DS$K_WARNING	=00000000	141 (1)	
DS$M_ABRTFLG	=00000040	147 (1)	
DS$M_BADTIME	=00100000	147 (1)	

```
DS$M_BATCH                      =00400000       147     (1)
DS$M_BRKCLR                     =00001000       147     (1)
DS$M_BRKPT                      =00000800       147     (1)
DS$M_CHARFLG                    =00000100       147     (1)
DS$M_CMDFLG                     =00000080       147     (1)
DS$M_CTRLC                      =00000001       147     (1)
DS$M_CTRLO                      =00010000       147     (1)
DS$M_DEVFLG                     =00000200       147     (1)
DS$M_DISABLCC                   =01000000       147     (1)
DS$M_DONFLG                     =00002000       147     (1)
DS$M_ERRFLG                     =00000010       147     (1)
DS$M_EXCEPT                     =00080000       147     (1)
DS$M_EXETST                     =00040000       147     (1)
DS$M_HLTFLG                     =00000008       147     (1)
DS$M_LODFLG                     =00000002       147     (1)
DS$M_MEMMGT                     =00008000       147     (1)
DS$M_OUTPUT                     =00800000       147     (1)
DS$M_RUBFLG                     =00000020       147     (1)
DS$M_SCRIPT                     =00200000       147     (1)
DS$M_SETIMR                     =02000000       147     (1)
DS$M_STRFLG                     =00000004       147     (1)
DS$M_SUBT                       =00004000       147     (1)
DS$M_SYSFLG                     =00000400       147     (1)
DS$M_TIMRON                     =00020000       147     (1)
DS$SOFTIPL5                      00000000-XR                    641     (1)
DS$V_ABRTFLG                    =00000006       147     (1)
DS$V_BADTIME                    =00000014       147     (1)
DS$V_BATCH                      =00000016       147     (1)
DS$V_BRKCLR                     =0000000C       147     (1)
DS$V_BRKPT                      =0000000B       147     (1)
DS$V_CHARFLG                    =00000008       147     (1)
DS$V_CMDFLG                     =00000007       147     (1)
DS$V_CTRLC                      =00000000       147     (1)
DS$V_CTRLO                      =00000010       147     (1)
DS$V_DEVFLG                     =00000009       147     (1)
DS$V_DISABLCC                   =00000018       147     (1)
DS$V_DONFLG                     =0000000D       147     (1)
DS$V_ERRFLG                     =00000004       147     (1)
DS$V_EXCEPT                     =00000013       147     (1)
DS$V_EXETST                     =00000012       147     (1)
DS$V_HLTFLG                     =00000003       147     (1)
DS$V_LODFLG                     =00000001       147     (1)
DS$V_MEMMGT                     =0000000F       147     (1)
DS$V_OUTPUT                     =00000017       147     (1)
DS$V_RUBFLG                     =00000005       147     (1)
DS$V_SCRIPT                     =00000015       147     (1)
DS$V_SETIMR                     =00000019       147     (1)
DS$V_STRFLG                     =00000002       147     (1)
DS$V_SUBT                       =0000000E       147     (1)
DS$V_SYSFLG                     =0000000A       147     (1)
DS$V_TIMRON                     =00000011       147     (1)     #-290   (1)     #-327   (1)     #-415   (1)     #-541   (1)
DS$_ARITH                       =006600D0       141     (1)
DS$_ASBE                        =00660118       141     (1)
DS$_BADLINK                     =006600F0       141     (1)
DS$_BADTYPE                     =006600E8       141     (1)
DS$_BIIC                        =00660120       141     (1)
DS$_CHME                        =006600A8       141     (1)
```

```
DS$_CHMK                        =006600E0       141     (1)
DS$_DEVNAME                     =00660108       141     (1)
DS$_ERROR                       =00660002       141     (1)
DS$_FHWE                        =00660068       141     (1)
DS$_FRAGBUF                     =00660080       141     (1)
DS$_GK_EXE_MCHK                 =00000000       148     (1)
DS$_GK_INITSCB                  =00000002       148     (1)     #-340     (1)
DS$_GK_INT_WORK                 =00000003       148     (1)
DS$_GK_TST$MCHK                 =00000001       148     (1)
DS$_ICBUSY                      =006600C8       141     (1)     #-417     (1)
DS$_ICERR                       =006600C0       141     (1)
DS$_IHWE                        =00660060       141     (1)
DS$_ILLCHAR                     =00660018       141     (1)
DS$_ILLPAGCNT                   =00660078       141     (1)
DS$_ILLUNIT                     =00660100       141     (1)
DS$_INSFMEM                     =00660050       141     (1)
DS$_IPL2HI                      =006600B8       141     (1)
DS$_IVADDR                      =00660040       141     (1)     #-392     (1)
DS$_IVVECT                      =00660038       141     (1)     #-591     (1)
DS$_KRNLSTK                     =00660090       141     (1)
DS$_LOGIC                       =00660070       141     (1)
DS$_MCHK                        =00660088       141     (1)
DS$_MMOFF                       =00660058       141     (1)
DS$_NEEDUNIT                    =006600F8       141     (1)
DS$_NODE                        =00660128       141     (1)
DS$_NOPCS                       =00660110       141     (1)
DS$_NORMAL                      =00660001       141     (1)     #-465     (1)
DS$_NOSUPPORT                   =006600B1       141     (1)
DS$_NOTDON                      =00660030       141     (1)
DS$_NOTIMP                      =006600B0       141     (1)       141     (1)
DS$_NULLSTR                     =00660010       141     (1)
DS$_OVERFLOW                    =00660008       141     (1)
DS$_POWER                       =00660098       141     (1)
DS$_PROGERR                     =00660020       141     (1)
DS$_SEVERE                      =00660004       141     (1)
DS$_TRANSL                      =006600A0       141     (1)
DS$_TRUNCATE                    =00660028       141     (1)
DS$_UNEXPINT                    =006600D8       141     (1)
DS$_VASFULL                     =00660048       141     (1)
DS$_WARNING                     =00660000       141     (1)       141     (1)
DSI$CHANNEL_VEC                  00000000-XR                      349     (1)
DSI$TIMER                        00000000-XR                      643     (1)
DSI$TIMSRV                       00000000-XR                      239     (1)
DSR$FAULT_CLEAR                  00000000-XR                      341     (1)
DSX$CLRVEC                       00000192-R      504     (1)
DSX$CLRVEC_X                     00000214-R      545     (1)
DSX$INITSCB                      00000000-R      285     (1)
DSX$SETIPL                       00000238-R      615     (1)
DSX$SETVEC                       000000D7-R      388     (1)
DSX$SETVEC_X                     0000018B-R      467     (1)     #-419     (1)
EXE$FORKDSPTH                    00000000-XR                      642     (1)     644     (1)     645     (1)     646     (1)
                                                                  647     (1)
EXE$GL_SCB                       0000000D-R      172     (1)     #-339     (1)
EXE_ACCVIO                       00000000-XR                      199     (1)
EXE_ARITH                        00000000-XR                      204     (1)
EXE_BREAK                        00000000-XR                      202     (1)
EXE_CHME                         00000000-XR                      208     (1)
```

```
EXE_CHMS                        00000000-XR                    209     (1)
EXE_CHMU                        00000000-XR                    210     (1)
EXE_CMODKRNL                    00000000-XR                    207     (1)
EXE_COMPAT                      00000000-XR                    203     (1)
EXE_KRNLSTK                     00000000-XR                    193     (1)
EXE_MCHK                        00000000-XR                    192     (1)
EXE_OPCCUS                      00000000-XR                    196     (1)
EXE_OPCDEC                      00000000-XR                    195     (1)
EXE_POWER                       00000000-XR                    194     (1)
EXE_RADRMOD                     00000000-XR                    198     (1)
EXE_ROPRAND                     00000000-XR                    197     (1)
EXE_TBIT                        00000000-XR                    201     (1)
EXE_TRANSL                      00000000-XR                    200     (1)
EXE_UNEXPINT                    00000000-XR                    350     (1)     668     (1)
ICCS$M_ERR                      =80000000       151     (1)  #-430     (1)
ICCS$M_INT                      =00000080       152     (1)  #-431     (1)
IOC$IOPOST                      00000000-XR                    640     (1)
IS                              =00000001       149     (1)  192     (1)     193     (1)     194     (1)     239     (1)
                                                             317     (1)     532     (1)
KB_CHECK                        00000000-XR                    292     (1)     468     (1)     546     (1)
ONLY_SCB                        00000000-XR                    320     (1)
PR$_ICCS                        =00000018                      #-432   (1)   #-434   (1)
PR$_IPL                         =00000012                      #-301   (1)   #-343   (1)   #-622   (1)   #-657   (1)
PR$_SCBB                        =00000011                      #-338   (1)
PSL$S_IPL                       =00000005                      #-624   (1)
PSL$V_IPL                       =00000010                      #-623   (1)
PSL$V_IS                        =0000001A       146     (1)  #-287     (1)   #-617     (1)
SCB$L_BREAK                     0000002C                       #-403   (1)   #-513     (1)
SCB$L_CHMK                      00000040                       #-398   (1)   #-508     (1)
SCB$L_SFTLVL1                   00000084                       #-441   (1)   #-523     (1)
SCB$L_SFTLVL15                  000000BC                       #-443   (1)   #-525     (1)
SCB$L_TBIT                      00000028                       #-408   (1)   #-518     (1)
SCB$L_TIMER                     000000C0                       #-413   (1)   #-538     (1)
SCBVECTOR_000                   00000000-XR                    191     (1)
SCBVECTOR_038                   00000000-XR                    205     (1)
SCBVECTOR_03C                   00000000-XR                    206     (1)
SCBVECTOR_050                   00000000-XR                    211     (1)
SCBVECTOR_054                   00000000-XR                    212     (1)
SCBVECTOR_058                   00000000-XR                    213     (1)
SCBVECTOR_05C                   00000000-XR                    214     (1)
SCBVECTOR_060                   00000000-XR                    215     (1)
SCBVECTOR_064                   00000000-XR                    216     (1)
SCBVECTOR_068                   00000000-XR                    217     (1)
SCBVECTOR_06C                   00000000-XR                    218     (1)
SCBVECTOR_070                   00000000-XR                    219     (1)
SCBVECTOR_074                   00000000-XR                    220     (1)
SCBVECTOR_078                   00000000-XR                    221     (1)
SCBVECTOR_07C                   00000000-XR                    222     (1)
SCBVECTOR_080                   00000000-XR                    223     (1)
SCBVECTOR_0C4                   00000000-XR                    240     (1)
SCBVECTOR_0C8                   00000000-XR                    241     (1)
SCBVECTOR_0CC                   00000000-XR                    242     (1)
SCBVECTOR_0D0                   00000000-XR                    243     (1)
SCBVECTOR_0D4                   00000000-XR                    244     (1)
SCBVECTOR_0D8                   00000000-XR                    245     (1)
SCBVECTOR_0DC                   00000000-XR                    246     (1)
SCBVECTOR_0E0                   00000000-XR                    247     (1)
```

```
SCBVECTOR_0E4                   00000000-XR                    248    (1)
SCBVECTOR_0E8                   00000000-XR                    249    (1)
SCBVECTOR_0EC                   00000000-XR                    250    (1)
SCBVECTOR_0F0                   00000000-XR                    251    (1)
SCBVECTOR_0F4                   00000000-XR                    252    (1)
SCBVECTOR_0F8                   00000000-XR                    253    (1)
SCBVECTOR_0FC                   00000000-XR                    254    (1)
SCB_ACCVIO                      00000020-R    199    (1)
SCB_ARITH                       00000034-R    204    (1)
SCB_BASE                        00000000-XR                    303    (1)      337    (1)      396    (1)      462    (1)
                                                             #-537    (1)    #-585    (1)
SCB_BREAK                       0000002C-R    202    (1)
SCB_CHME                        00000044-R    208    (1)
SCB_CHMK                        00000040-R    207    (1)
SCB_CHMS                        00000C48-R    209    (1)
SCB_CHMU                        0000004C-R    210    (1)
SCB_COMPAT                      00000030-R    203    (1)
SCB_IMAGE                       00000000-R    190    (1)      173    (1)      304    (1)    #-535    (1)
SCB_KRNLSTK                     00000008-R    193    (1)
SCB_MCHK                        00000004-R    192    (1)
SCB_OPCCUS                      00000014-R    196    (1)
SCB_OPCDEC                      00000010-R    195    (1)
SCB_POWER                       0000000C-R    194    (1)
SCB_RADRMOD                     0000001C-R    198    (1)
SCB_ROPRAND                     00000018-R    197    (1)
SCB_SFTLVL1                     00000084-R    224    (1)
SCB_SFTLVL10                    000000A8-R    233    (1)
SCB_SFTLVL11                    000000AC-R    234    (1)
SCB_SFTLVL12                    000000B0-R    235    (1)
SCB_SFTLVL13                    000000B4-R    236    (1)
SCB_SFTLVL14                    000000B8-R    237    (1)
SCB_SFTLVL15                    000000BC-R    238    (1)
SCB_SFTLVL2                     00000088-R    225    (1)
SCB_SFTLVL3                     0000008C-R    226    (1)
SCB_SFTLVL4                     00000090-R    227    (1)
SCB_SFTLVL5                     00000094-R    228    (1)
SCB_SFTLVL6                     00000098-R    229    (1)
SCB_SFTLVL7                     0000009C-R    230    (1)
SCB_SFTLVL8                     000000A0-R    231    (1)
SCB_SFTLVL9                     000000A4-R    232    (1)
SCB_TBIT                        00000028-R    201    (1)
SCB_TIMER                       000000C0-R    239    (1)
SCB_TRANSL                      00000024-R    200    (1)
SCB_UNKINT                      00000000-XR                    309    (1)    #-318    (1)    #-348    (1)      532    (1)
                                                           #-585    (1)
SCH$ASTDFL                      00000000-XR                    638    (1)
SETVEC$_CODE                    =0000000C     145    (1)
SETVEC$_NARGS                   =00000003     145    (1)
SETVEC$_SRVADR                  =00000008     145    (1)
SETVEC$_VECTOR                  =00000004     145    (1)
SIZ...                          =00000001     147    (1)      147    (1)
SOFT$INT                        000002A0-R    655    (1)      224    (1)      225    (1)      226    (1)      227    (1)
                                                           228    (1)      229    (1)      230    (1)      231    (1)
                                                           232    (1)      233    (1)      234    (1)      235    (1)
                                                           236    (1)      237    (1)      238    (1)
SOFT$INT_A                      00000260-R    635    (1)  #-662    (1)    #-673    (1)
SS$_NORMAL                      00000000-XR                  #-289    (1)    #-543    (1)    #-618    (1)
```

UNKNOWN_INT 000000C4-R 347 (1) 310 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	148 (1)	
$DEFINI	1	142 (1)	142 (1) 143 (1) 144 (1)
$DS_CVTREG_DEF	1	140 (1)	140 (1)
$DS_DSDEF	2	141 (1)	141 (1)
$DS_SCBDEF	3	144 (1)	144 (1)
$DS_SETVEC_DEF	1	145 (1)	145 (1)
$EQU	1	148 (1)	141 (1) 146 (1) 148 (1)
$EQULS1	1	148 (1)	141 (1) 146 (1) 148 (1)
$EQULST	1	141 (1)	141 (1) 146 (1) 148 (1)
$GBLINI	2	141 (1)	141 (1) 146 (1) 147 (1) 148 (1)
$OFFDEF	1	140 (1)	140 (1) 145 (1)
$PRDEF	4	142 (1)	142 (1)
$PSLDEF	2	143 (1)	143 (1)
$VIELD	1	147 (1)	147 (1)
$VIELD1	1	148 (1)	147 (1)
CMK	1	287 (1)	287 (1) 617 (1)
CMKDEF	1	146 (1)	146 (1)
DSBINT	1	301 (1)	301 (1)
DSFDEF	3	147 (1)	147 (1)
ENBINT	1	343 (1)	343 (1)
FLTCLR_SEL	1	148 (1)	148 (1)
MODNAM	1	181 (1)	181 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.14	00:00:00.42
Command processing	138	00:00:00.89	00:00:01.98
Pass 1	608	00:00:08.57	00:00:16.25
Symbol table sort	0	00:00:00.66	00:00:01.62
Pass 2	165	00:00:01.97	00:00:02.65
Symbol table output	37	00:00:00.24	00:00:00.32
Object synopsis output	8	00:00:00.03	00:00:00.04
Cross-reference output	60	00:00:01.11	00:00:01.67
Assembler run totals	1052	00:00:13.63	00:00:24.95

The working set limit was 1000 pages.
40010 bytes (79 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 414 non-local and 31 local symbols.
683 source lines were read in Pass 1, producing 0 object records in Pass 2.
63 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	4
DRB1:[DS.WORK]DS.MLB;218	5
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	2
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	19

417 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) SCB/UPDA=(SCB.UPD,SCB.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT]DIAG/
```

Table of contents

(1) 108 Libraries and External Symbols ; [12]
(1) 118 External Symbol Declarations
(1) 141 Macros and Equated Symbols
(1) 160 Data Psect Declarations
(1) 194 Work Psect Declarations
(1) 212 Absolute Psect Declarations
(1) 226 SCRIPT$INIT - Initialize script
(1) 268 SCRIPT$OPEN - Initialize a script file
(1) 425 SCRIPT$FLUSH, STOP, CONT - Terminate all scripts open
(1) 489 SCRIPT$FINISH - Terminate script input file
(1) 544 DS_GETLINE - Get program data input line
(1) 965 COPY_IT - Copy and translate
(1) 1021 ADVANCE - Advance to next record in buffer

```
0000     1          .Title  SCRIPT  *** SCRIPT Read from command stream
0000     2          .Ident  /07-18/
0000     3          .NoShow Conditionals    ; For all the conditionals in the RMS macros
0000     4          .DSABL  GBL
0000     5
0000     6 ;
0000     7 ; Copyright (c) 1977, 1981
0000     8 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     9 ;
0000    10 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    11 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    12 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    13 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    14 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    15 ; TERMS.  TITLE     ND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    16 ; REMAIN IN DEC.
0000    17 ;
0000    18 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    19 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    20 ; CORPORATION.
0000    21 ;
0000    22 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    23 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    24
0000    25 ;++
0000    26 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    27 ;
0000    28 ; ABSTRACT:
0000    29 ;
0000    30 ; ENVIRONMENT:
0000    31 ;
0000    32 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    33 ;
0000    34 ; MODIFICATIONS:
0000    35 ;                Roger Riggs     20-Jun-78       Version 02.
0000    36 ;                True implementation of scripting in USER mode.
0000    37 ;       01       Added code to implement "skipping... " for mismatched scripts
0000    38 ;                Roger Riggs     12-Jun-1979
0000    39 ;       02       changed W^ to '^' on DSA$GL_FLAGS references
0000    40 ;                Roger Riggs, 17-Mar-1980, Version 5.3
0000    41 ;       03       DS$LOAD returns changed, reflected by getting SS$_NORMAL
0000    42 ;                Even if file was truncated.
0000    43 ;                Dave Butenhof   28-apr-80
0000    44 ;       04       Use $GET and $PUT for input/output if DS run from command file
0000    45 ;                Roger Riggs, 3-Sep-1980, Version 6.0
0000    46 ;       05       Modified SCRIPT$OPEN to use RMS instead of load
0000    47 ;
0000    48 ;                Dave Butenhof, 10-sep-1980, Version 6.0
0000    49 ;       06       Fix user mode RMS GETLINE code to perform case translation
```

```
0000     51 ;        07      - Jack Stansbury, 22-Oct-1981, Version 6.5
0000     52 ;                Fixed truncation errors. Also added .LIBRARY statements for
0000     53 ;                $DS and $DIAG.
0000     54 ;
0000     55 ;        08      - Dave Butenhof, 12-Nov-1981, version 6.5
0000     56 ;                Implement SET VERIFY/CLEAR VERIFY function.  When cleared,
0000     57 ;                this flag will cause command script input not to be echoed.
0000     58 ;                This is basically for the System Functional Text (EVXBB).
0000     59 ;
0000     60 ;        09      - Dave Butenhof, 17-Nov-1981, version 6.5
0000     61 ;                Implement type code bytes for all prompts.  The type code
0000     62 ;                will not be used in comparison of script file prompts,
0000     63 ;                but will be typed out to console or log file, if BINARY
0000     64 ;                flag is set.
0000     65 ;
0000     66 ;        10      -Jack Stansbury, 18-Dec-1981, Version 6.6
0000     67 ;                Added a little code in SCRIPT$FLUSH that will clear out the
0000     68 ;                QA bit in the DSA flags. This is to insure that QA gets turned
0000     69 ;                off when a Control-C interrupts a diagnostic in a script file,
0000     70 ;                and a non-continuable command (i.e., one that causes the script
0000     71 ;                file to be flushed) is issued. Before, if a START command was
0000     72 ;                issued, and a Control-C was typed, and a subsequent START
0000     73 ;                command was typed, the second one could be screwed up.
0000     74 ;
0000     75 ;        11      -Jack Stansbury, 26-Feb-1982, Version 6.6
0000     76 ;                Added comments to the DS_GETLINE routine to aid in
0000     77 ;                deciphering it.
0000     78 ;
0000     79 ;        12      - Jack Stansbury, 18-Mar-1982, Version 6.?
0000     80 ;                Took out any reference to the TypMsg routine. This is so
0000     81 ;                that type-coding can be added. Also added some new DS macros
0000     82 ;                that should make this more understandable.
0000     83 ;
0000     84 ;        13      Marion Baggett, 31-Mar-1982, Version 6.7
0000     85 ;                Added SYS$LIBRARY:LIB/
0000     86 ;
0000     87 ;        14      Marion Baggett, 8-Apr-1982, Version 6.7
0000     88 ;                Added type coding.
0000     89 ;
0000     90 ;        15      Jack Stansbury, 12-Apr-1982, Version 6.7
0000     91 ;                Added support for the Customer Runnable Diagnostics package.
0000     92 ;
0000     93 ;        16      John Ciukaj,    4-Apr-1983,     version 6.11
0000     94 ;                - Redefined reference to CRD bits in DSA Flags to
0000     95 ;                  distinguish between online and offline
0000     96 ;
0000     97 ;        17      Peter Green,    20-Jul-1983,    version 6.12
0000     98 ;                Added code to support scripting where prompts are
0000     99 ;                printed to the terminal and an answer is given by
0000    100 ;                the user. This is only done when the script prompt
0000    101 ;                ends with "\\".
0000    102 ;
0000    103 ;        18      M. Baggett      25-Jan-1984     Version 6.14
0000    104 ;                Changed dynamic allocation of XAB, FAB, RAB to static form
0000    105 ;                so script files will work.
0000    106 ;--
```

```
                          0000   108          .SBTTL  Libraries and External Symbols  ;                                    [12]
                          0000   109 ;
                          0000   110 ; INCLUDE FILES:
                          0000   111 ;
                          0000   112          .LIBRARY        /SYS$LIBRARY:LIB/       ;                                    [13]
                          0000   113          .Library        /$CRD/                  ;                                    [15]
                          0000   114          .LIBRARY        /$DS/                   ;                                    [07]
                          0000   115          .LIBRARY        /$DIAG/                 ;                                    [07]
                          0000   116
```

```
                     0000   118          .SBTTL  External Symbol Declarations
                     0000   119
                     0000   120 ;
                     0000   121 ; EXTERNAL SYMBOLS
                     0000   122 ;
                     0000   123          .EXTRN  DS$CLI,DSX$PRINT , DS_ERRSUP, DS$LOAD          ;                [12]
                     0000   124          .EXTRN  DSR$COMPLETION, DS$ABORT
                     0000   125          .EXTRN  SCAN$COMPARE, SCAN$SPACES
                     0000   126          .EXTRN  DS$GW_TTOUT, DS$GW_TTIN
                     0000   127          .EXTRN  DS$GL_CLIBASE
                     0000   128          .EXTRN  SYS$FAO, SYS$QIOW, SYS$QIO, SYS$WAKE, SYS$HIBER, SYS$READEF
                     0000   129          .EXTRN  SS$_NORMAL, SS$_ENDOFFILE, SS$_CONTROLC, SS$_ILLIOFUNC
                     0000   130          .EXTRN  IO$_READPROMPT ,IO$_WRITEVBLK, IO$_READVBLK
                     0000   131          .EXTRN  DS$GL_FLAGS
                     0000   132          .EXTRN  EXE$ALONONPAGED, EXE$DEANONPAGED
                     0000   133          .EXTRN  DS$K_BUFSIZ, DS$GT_BUFFER
                     0000   134          .EXTRN  QIO$CLEANUP, INIT_CONTEXT, BEGIN
                     0000   135          .EXTRN  DS$RAB_INPUT, DS$RAB_OUTPUT
                     0000   136          .EXTRN  DSX$TYPE_OUT                ;                            [09]
                     0000   137          .EXTRN  DEBUG_THE_SUCKER            ;                            [99]
                     0000   138          .Extrn  DS$GA_CRD_DS_Interface      ; The address of the routine to ...  [15]
                     0000   139                                              ; ... communicate with CRD       [15]
```

```
                    0000   141          .SBTTL Macros and Equated Symbols
                    0000   142 ;
                    0000   143 ; EQUATED SYMBOLS:
                    0000   144 ;
                    0000   145
00000400            0000   146          SCRIPT$MINCORE == 2*512          ; NEEDS AT LEAST TWO PAGES FOR SCRIPT   [12]
                    0000   147
                    0000   148          $DS_DSADEF
                    0000   149          $DS_DSDEF
                    0000   150          DSFDEF
                    0000   151          CLIDEF
                    0000   152          $FABDEF
                    0000   153          $RMSDEF
                    0000   154          $RABDEF
                    0000   155          $XABDEF
                    0000   156          $XABFHCDEF
                    0000   157          $DS_TypeDef                      ; Define DS error codes                 [09]
                    0000   158          $CRD_Literals                    ; define some CRD equated symbols       [15]
```

```
                                0000   160          .SBTTL Data Psect Declarations
                           00000000   161          .Psect  Data, Shr, Noexe, Ncwrt, Byte
                                0000   162 ;
                                0000   163 ; STORAGE:
                                0000   164 ;
                                0000   165          MODNAM  SCRIPT
                                0007   166
                                0007   167 T_DEF_COM:
                     4D 4F 43 2E  0007   168          .ASCII  ".COM"
                                000B   169
                                000B   170 T_NULL:
                              00'  000B   171          .ASCIC  ""
                              00   000B
                                000C   172
                                C00C   173 T_EOF:
           3E 46 4F 45 3C 20 40 00'  000C   174          .ASCIC  "@ <EOF>"
                              07   000C
                                0014   175
                                0014   176 T_PROMPT:
        2F 21 44 41 21 43 41 21 00'  0014   177          .ASCIC  "!AC!AD!/"
                              08   0014
                                001D   178
                                001D   179 T_PROMPT2:
           43 41 21 43 41 21 00'  001D   180          .ASCIC  "!AC!AC"
                              06   001D
                                0024   181
                                0024   182 T_PROMPT3:
              2F 21 53 41 21 00'  0024   183          .ASCIC  "!AS!/"
                              05   0024
                                002A   184
                                002A   185 T_SKIP:
2E 2E 20 67 6E 69 70 70 69 6B 53 00'  002A   186          .ASCIC  'Skipping ..."!AD"!/'
                 2F 21 22 44 41 21 22 2E  0036
                                13   002A
                                003E   187
                                003E   188 T_ECHO:
              2F 21 44 41 21 53 41 21 00'  003E   189          .ASCIC  '!AS!AD!/'      ; Format line echo to log file
                              08   003E
                                0047   190
                                0047   191 T_PNF:
43 41 21 22 20 74 70 6D 6F 72 50 00'  0047   192          .ASCIC  'Prompt "!AC" not found in script!/'
20 64 6E 75 6F 66 20 74 6F 6E 20 22   0053
   2F 21 74 70 69 72 63 73 20 6E 69   005F
                                22   0047
```

```
                     006A   194          .SBTTL  Work Psect Declarations
                 00000000   195          .PSECT  Work, Noshr, Noexe, Wrt, Long
                     0000   196 SCRIPT_XAB:     $XABFHC             ; XAB                                          [18]
                     002C   197 SCRIPT_FAB:     $FAB    XAB=SCRIPT_XAB  ; FAB                                      [18]
                     007C   198 SCRIPT_RAB:     $RAB    FAB=SCRIPT_FAB  ;                                          [18]
                     00C0   199
                     00C0   200 L_LINK:
                000000C4  00C0   201          .BLKL   1               ; ADDRESS (IF ANY) OF CURRENT SCRIPT DESCRIPTOR
                     00C4   202
                     00C4   203 L_FLAGS:
                000000C8  00C4   204          .BLKL   1               ; FLAG BITS
                     00C8   205
                00000001  00C8   206          V_ECHOED=1              ; SET IF LINE HAS BEEN READ BUT NOT ECHOED
                     00C8   207
                     C0C8   208 DS$GB_QIOACT::
                       00   00C8   209          .BYTE   0               ; Low Bit set if input QIO active
                     00C9   210                                         ; Used by PRINTx in user mode to cancel input
```

```
                       00C9   212          .SBTTL Absolute Psect Declarations
                       00C9   213          .SAVE
                       00C9   214          .PSECT  $ABS$, ABS
                00000000 0000 215 .=0
                00000000 0000 216 SC$L_LINK:       .BLKL   1       ; ADDRESS (IF ANY) OF NEXT STACKED SCRIPT
                00000004 0004 217                  .BLKL   1       ; UNUSED
                00000008 0008 218 SC$W_SIZE:       .BLKW   1       ; SIZE OF ALLOCATED SPACE
                0000000A 000A 219 SC$W_LEN:        .BLKW   1       ; Length of current record SC$W_CURR
                0000000C 000C 220 SC$W_CURR:       .BLKW   1       ; OFFSET TO CURRENT RECORD IN BUFFER
                0000000E 000E 221 SC$W_END:        .BLKW   1       ; NUMBER OF BYTES IN THIS BUFFER
                00000010 0010 222 SC$A_BUF:        .BLKB   0       ; OFFSET TO START OF TEXT
                00000010 0010 223 SC$C_BLN=.
                       00C9   224          .RESTORE
```

```
                     00C9   226          .SBTTL  SCRIPT$INIT -   Initialize script
                 00000000   227          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0000   228 ;++
                     0000   229 ; FUNCTIONAL DESCRIPTION:
                     0000   230 ;
                     0000   231 ;       This routine is called from KERNEL during initialization to
                     0000   232 ;       initialize the SCRIPT functionality.
                     0000   233 ;
                     0000   234 ; CALLING SEQUENCE:
                     0000   235 ;
                     0000   236 ;       BSBW    SCRIPT$INIT
                     0000   237 ;
                     0000   238 ; INPUT PARAMETERS:
                     0000   239 ;
                     0000   240 ;       NONE
                     0000   241 ;
                     0000   242 ; IMPLICIT INPUTS:
                     0000   243 ;
                     0000   244 ;       NONE
                     0000   245 ;
                     0000   246 ; OUTPUT PARAMETERS:
                     0000   247 ;
                     0000   248 ;       NONE
                     0000   249 ;
                     0000   250 ; IMPLICIT OUTPUTS:
                     0000   251 ;
                     0000   252 ;       NONE
                     0000   253 ;
                     0000   254 ; COMPLETION CODES:
                     0000   255 ;
                     0000   256 ;       NONE
                     0000   257 ;
                     0000   258 ; SIDE EFFECTS:
                     0000   259 ;
                     0000   260 ;       NONE
                     0000   261 ;--
```

```
                        0000   263 SCRIPT$INIT::
000000C0'EF    D4       0000   264          CLRL    L_LINK                  ; Clear link to first script
       0163    30       0006   265          BSBW    SCRIPT$STOP             ; Indicate no scripts active          [11]
               05       0009   266          RSB                             ; Return
```

```
                     000A   268          .SBTTL  SCRIPT$OPEN -   Initialize a script file
                0000000A    269          .Psect  Code, Shr, Exe, Nowrt, Byte
                     000A   270 ;++
                     000A   271 ; FUNCTIONAL DESCRIPTION:
                     000A   272 ;
                     000A   273 ;       This routine is called from CLI to process a @filespec commard.
                     000A   274 ;       It allocates space for the necessary internal data structures,
                     000A   275 ;       Open the file specified and sets a flag that input should be
                     000A   276 ;       taken from the script file instead of the console.
                     000A   277 ;
                     000A   278 ; CALLING SEQUENCE:
                     000A   279 ;
                     000A   280 ;       BSBW    SCRIPT$OPEN
                     000A   281 ;
                     C00A   282 ; INPUT PARAMETERS:
                     000A   283 ;
                     000A   284 ;       NONE
                     000A   285 ;
                     000A   286 ; IMPLICIT INPUTS:
                     000A   287 ;
                     000A   288 ;       CLI$GL_BASE + CLI$Q_FILE        Length and Address of filespec
                     000A   289 ;
                     000A   290 ; OUTPUT PARAMETERS:
                     000A   291 ;
                     000A   292 ;       NONE
                     000A   293 ;
                     000A   294 ; IMPLICIT OUTPUTS:
                     000A   295 ;
                     000A   296 ;       DS$V_SCRIPT                     May be set in DS$GL_FLAGS
                     000A   297 ;
                     000A   298 ; COMPLETION CODES:
                     000A   299 ;
                     000A   300 ;       RMS Standard completion codes
                     000A   301 ;
                     000A   302 ; SIDE EFFECTS:
                     000A   303 ;
                     000A   304 ;       NONE
                     000A   305 ;--
```

```
                                 000A   307 SCRIPT$OPEN::
                          0144   30  000A   308         BSBW    SCRIPT$FLUSH            ; Flush scripts if console command      [11]
                00000000'EF      16  000D   309         JSB     INIT_CONTEXT            ; Refresh Stacks, AST,PCB,PHD etc       [07]
                                 0013   310         Br_If_User      10$                 ; If user mode, branch                  [12]
        00000000'EF   6C    FA   001B   311         CALLG   (AP),QIO$CLEANUP            ; Flush current QIO database
                                 0022   312
                                 0022   313 ;
                                 0022   314 ; Set pointer to XAB
                                 0022   315 ;
                                 0022   316
    57  00000000'EF    9E       0022   317 10$:    MOVAB   SCRIPT_XAB,R7               ; SAVE ADDRESS OF XAB                   [18]
                                 0029   318 ;
                                 0029   319 ; Fill in FAB
                                 0029   320 ;
    59  0000002C'EF    9E       C029   321         MOVAB   SCRIPT_FAB,R9               ; SAVE ADDRESS OF FAB                   [18]
    52  00000000'EF    9E       0030   322         MOVAB   DS$GL_CLIBASE, R2           ; Address of CLI database               [11]
       34 A9    08 A2    90     0037   323         MOVB    CLI$Q_FILE (R2), -          ; File name size                        [11]
                                 003C   324           FAB$B_FNS (R9)                ;                                       [11]
       2C A9    0C A2    D0     003C   325         MOVL    CLI$Q_FILE+4 (R2), -        ; File name address                     [11]
                                 0041   326           FAB$L_FNA (R9)                ;                                       [11]
         35 A9    04     90     0041   327         MOVB    #4, FAB$B_DNS (R9)          ; Default name size                     [11]
30 A9   00000007'EF    9E       0045   328         MOVAB   L^T_DEF_COM, -              ; Default name address                  [07]
                                 004D   329           FAB$L_DNA (R9)                ;                                       [11]
```

```
                                004D   331 ;+
                                004D   332 ; Set pointer to RAB
                                004D   333 ;-
            58   0000007C'EF   9E  004D   334         MOVAB   SCRIPT_RAB,R8           ; SAVE ADDRESS OF RAB                   [18]
                                0054   335 ;+
                                0054   336 ; $OPEN the file
                                0054   337 ;-
                                0054   338
                                0054   339         $OPEN   FAB = (R9)
                     FFA0'      30  005D   340         BSBW    DSR$COMPLETION          ; Print errors if any
                    03 50       E8  0060   341         BLBS    R0,70$                  ; Branch to continue
                     00E5       31  0063   342         BRW     SCRIPT$OPEN_X           ; Exit w/error
                                0066   343
                                0066   344 ;+
                                0066   345 ; Allocate buffer for script
                                0066   346 ;-
                                0066   347
           50    10 A7    09    78  0066   348 70$:    ASHL    #9,XAB$L_EBK(R7),R0     ; Get length of file
                51    14 A7    3C  006B   349         MOVZWL  XAB$W_FFB(R7),R1        ; Get first free byte
                51    10 A041    9E  006F   350         MOVAB   SC$C_BLN(R0)[R1],R1     ; Length required for script + a little
                     FF89'      30  0074   351         BSBW    EXE$ALONONPAGED         ; Allocate space for buffer
                    03 50       E8  0077   352         BLBS    R0, 71$                 ; Branch around if no error             [12]
                     0081       31  007A   353         BRW     160$                    ; Branch if error, Close and exit       [12]
                                007D   354
                        62      7C  007D   355 71$:    CLRQ    (R2)                    ; Clear first part of SC$...            [12]
                     08 A2      7C  007F   356         CLRQ    8(R2)                   ; Clear second part of SC$...
                  5B    52      D0  0082   357         MOVL    R2,R11                  ; base script control area
               08 AB    51      B0  0085   358         MOVW    R1,SC$W_SIZE(R11)       ; Set the length allocated
         6B    000000C0'EF      D0  0089   359         MOVL    L^L_LINK,SC$L_LINK(R11) ; Make current script the previous     [07]
000000C0'EF    6B               DE  0090   360         MOVAL   (R11),L^L_LINK          ; Make this the current script          [07]
                                0097   361         Clear_StrFlg                    ; Clear started flag                    [12]
                                009F   362         Clear_CtrlC                     ; Clear ^C flag                         [12]
            53    10 AB    9E  00A7   363         MOVAB   SC$A_BUF(R11),R3        ; Address of free space in buffer
         54    51    10    C3  00AB   364         SUBL3   #SC$A_BUF,R1,R4         ; Length left in buffer
```

```
                                00AF   366 ;+
                                00AF   367 ; Connect to record stream
                                00AF   368 ;-
                                00AF   369
                                00AF   370          $CONNECT RAB = (R8)
                        43 50    E9    00B8   371          BLBC    R0,160$                     ; Branch if error, Close and exit
                                00BB   372
                  24 A8   01 A3  9E    00BB   373 100$:    MOVAB   1(R3), RAB$L_UBF(R8)        ; Set address to put data
                  20 A8   FF 8F  9B    00C0   374          MOVZBW  #255, RAB$W_USZ(R8)         ; Limit size to 255 characters
               54   00000100 8F  D1    00C5   375          CMPL    #256, R4                    ; Room for count byte and 255 characters?
                           05  15    00CC   376          BLEQ    110$                        ; Branch if there is
                20 A8   54   01  A3    00CE   377          SUBW3   #1, R4, RAB$W_USZ(R8)       ; Shorten transfer size
                                00D3   378
                                00D3   379 110$:    $GET    RAB = (R8)                  ; $GET a record
                        13 50    E9    00DC   380          BLBC    R0, 130$                    ; Branch if error, includes EOF, RTB
                     50   22 A8  3C    00DF   381          MOVZWL  RAB$W_RSZ(R8), R0           ; Get actual record length
                        63   50  90    00E3   382          MOVB    R0, (R3)                    ; Set length of record
                  53   01 A340  9E    00E6   383          MOVAB   1(R3)[R0], R3               ; Set new free
                        54   50  C2    00EB   384          SUBL    R0, R4                      ; Decrement length remaining
                             54  D7    00EE   385          DECL    R4                          ; Less one for length
                             C9  11    00F0   386          BRB     100$                        ; Try again
```

```
                                00F2   388 ;+
                                00F2   389 ; Error, EOF, or Record too big on GET
                                00F2   390 ;-
                                00F2   391
            52   50   D0       00F2   392 130$:   MOVL    R0,R2                    ; Save return code
                                00F5   393         $DISCONNECT     RAB = (R8)
                                00FE   394
                                00FE   395 160$:   $CLOSE  FAB = (R9)
            50   52   D0       0107   396         MOVL    R2,R0                    ; Restore error code
 0001827A   8F   50   D1       010A   397         CMPL    R0, #RMS$_EOF            ; End of file?
                08   13       0111   398         BEQL    180$                     ; Branch if legit EOF
                FEEA'   30     0113   399         BSBW    DSR$COMPLETION           ; Report the error
                006F   30       0116   400         BSBW    SCRIPT$FINISH            ; Error, flush the script
                                0119   401
                30   11       C119   402 170$:   BRB     SCRIPT$OPEN_X            ; Exit
                                011B   403
                                011B   404 ;+
                                011B   405 ; EOF
                                011B   406 ;-
                                011B   407
            50   53   D0       011B   408 180$:   MOVL    R3, R0                   ; Free address in buffer
       51   50   5B   C3       011E   409         SUBL3   R11, R0, R1              ; Length used
  0E   AB   51   10   A3       0122   410         SUBW3   #SC$A_BUF, R1, SC$W_END(R11); Save length of data
       51   0F   A1   9E       0127   411         MOVAB   15(R1), R1               ; Round up to block boundry
  08   AB   51   0F   AB       012B   412         BICW3   #15, R1, SC$W_SIZE(R11)  ; Truncate to block boundry and save
            54   0F   AA       0130   413         BICW    #15, R4                  ; Length remaining to block boundry
                0E   13       0133   414         BEQL    190$                     ; None, finish
       50   0F   A0   9E       0135   415         MOVAB   15(R0), R0               ; Round up to block boundry
            50   0F   CA       0139   416         BICL    #15, R0                  ; Finish rounding
  08   A0   54   B0       013C   417         MOVW    R4, 8(R0)                ; Set size of section to be released
                FEBD'   30     0140   418         BSBW    EXE$DEANONPAGED          ; Release the unused portion
                                0143   419
                                0143   420 190$:   Set_Script                     ; Set to script mode and exit          [12]
                                0148   421
                                0148   422 SCRIPT$OPEN_X:
 00000000'EF    17   0148   423         JMP     BEGIN                    ; And force reentry of CLI
```

```
                     0151   425          .SBTTL  SCRIPT$FLUSH, STOP, CONT -      Terminate all scripts open
                 00000151   426          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0151   427 ;++
                     0151   428 ; FUNCTIONAL DESCRIPTION:
                     0151   429 ;
                     0151   430 ;       SCRIPT$FLUSH    Will SCRIPT$FINISH until L_LINK is zero.
                     0151   431 ;       SCRIPT$STOP     Clears DS$V_SCRIPT
                     0151   432 ;       SCRIPT$CONT     Sets DS$V_SCRIPT if L_LINK NEQ 0
                     0151   433 ;
                     0151   434 ; CALLING SEQUENCE:
                     0151   435 ;
                     0151   436 ;       BSBW    SCRIPT$FLUSH
                     0151   437 ;       BSBW    SCRIPT$STOP
                     0151   438 ;       BSBW    SCRIPT$CONT
                     0151   439 ;
                     0151   440 ; INPUT PARAMETERS:
                     0151   441 ;
                     0151   442 ;       NONE
                     0151   443 ;
                     0151   444 ; IMPLICIT INPUTS:
                     0151   445 ;
                     0151   446 ;       Current DS$GL_FLAGS and script OWN data
                     0151   447 ;
                     0151   448 ; OUTPUT PARAMETERS:
                     0151   449 ;
                     0151   450 ;       NONE
                     0151   451 ;
                     0151   452 ; IMPLICIT OUTPUTS:
                     0151   453 ;
                     0151   454 ;       NONE
                     0151   455 ;
                     0151   456 ; COMPLETION CODES:
                     0151   457 ;
                     0151   458 ;       SS$_NORMAL
                     0151   459 ;
                     0151   460 ; SIDE EFFECTS:
                     0151   461 ;
                     0151   462 ;       NONE
                     0151   463 ;--
```

```
                                0151   465 SCRIPT$FLUSH::
                                0151   466         Clear_QA                        ; Turn off the QA flag                  [12]
                                0159   467
                                0159   468 10$:    Br_If_Script    20$             ; Branch if script mode                 [12]
                          25    10    0161   469         BSBB    SCRIPT$FINISH   ; YES, FINISH TOP LEVEL                  [07]
                000000C0'EF     D5    0163   470         TSTL    L^L_LINK        ; ANY SCRIPTS ACTIVE?                   [07]
                          EE    12    0169   471         BNEQ    10$             ; AND TRY AGAIN                         [10]
                                0168   472
                          05    0168   473 20$:    RSB
                                016C   474
                                016C   475
                                016C   476 SCRIPT$STOP::
                                016C   477         Clear_Script                    ; Clear script flag                     [12]
                          05    0174   478         RSB                             ;                                       [12]
                                0175   479
                                0175   480
                                0175   481 SCRIPT$CONT::
                          F5    10    0175   482         BSBB    SCRIPT$STOP     ; Clear script flag
                000000C0'EF     D5    0177   483         TSTL    L^L_LINK        ; Any scripts present?                  [07]
                          08    13    017D   484         BEQL    10$             ; Branch if not
                                017F   485         Set_Script                      ; Set script flag                       [12]
                                0187   486
                          05    0187   487 10$:    RSB
```

```
                     0188    489          .SBTTL  SCRIPT$FINISH - Terminate script input file
                  00000188   490          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0188    491 ;++
                     0188    492 ; FUNCTIONAL DESCRIPTION:
                     0188    493 ;
                     0188    494 ;       This routine will $CLOSE the current script file (if any) and return the
                     0188    495 ;       space allocated. If this is the last script file, it will clear the
                     0188    496 ;       DS$V_SCRIPT flag in DS$GL_FLAGS.
                     0188    497 ;
                     0188    498 ; CALLING SEQUENCE:
                     0188    499 ;
                     0188    500 ;       BSBW    SCRIPT$FINISH
                     0188    501 ;
                     0188    502 ; INPUT PARAMETERS:
                     0188    503 ;
                     0188    504 ;       NONE
                     0188    505 ;
                     0188    506 ; IMPLICIT INPUTS:
                     0188    507 ;
                     0188    508 ;       Current DS$GL_FLAGS and Script's OWN data.
                     0188    509 ;
                     0188    510 ; OUTPUT PARAMETERS:
                     0188    511 ;
                     0188    512 ;       NONE
                     0188    513 ;
                     0188    514 ; IMPLICIT OUTPUTS:
                     0188    515 ;
                     0188    516 ;       DS$GL_SCRIPT and OWN DATA
                     0188    517 ;
                     0188    518 ; COMPLETION CODES:
                     0188    519 ;
                     0188    520 ;       SS$_NORMAL
                     0188    521 ;
                     0188    522 ; SIDE EFFECTS:
                     0188    523 ;
                     0188    524 ;       NONE
                     0188    525 ;--
```

```
                          0188   527 SCRIPT$FINISH:
                    3C    BB  0188   528         PUSHR   #^M<R2, R3, R4, R5>     ; Save registers
                                0188A  529
          50  000000C0'EF    D0  0188A  530 10$:    MOVL    L^L_LINK, R0            ; GET POINTER TO CURRENT SCRIPT          [07]
                    12    13  01891  531         BEQL    30$                     ; NONE, EXIT RIGHT AWAY
000000C0'EF    60    D0  01893  532         MOVL    SC$L_LINK(R0),L^L_LINK  ; MAKE NEXT CURRENT                      [07]
              FE63'   30  0189A  533         BSBW    EXE$DEANONPAGED         ; DEALLOCATE IT
                                0189D  534
     000000C0'EF    D5  0189D  535 20$:    TSTL    L^L_LINK                ; ANY CURRENT SCRIPT?                    [07]
                    02    12  018A3  536         BNEQ    40$                     ; YES, JUST EXIT
                                018A5  537
                    C5    10  018A5  538 30$:    BSBB    SCRIPT$STOP             ; Clear script active flag               [11]
                                018A7  539
                 50   00'    D0  018A7  540 40$:    MOVL    S^#SS$_NORMAL, R0       ; IT WORKED!
                    3C    BA  01AA   541         POPR    #^M<R2, R3, R4, R5>     ; Restore registers
                          05  01AC   542         RSB                             ; RETURN
```

```
                       C1AD   544          .SBTTL  DS_GETLINE -    Get program data input line
                   000001AD   545          .Psect  Code, Shr, Exe, Nowrt, Byte
                       01AD   546 ;++
                       01AD   547 ; FUNCTIONAL DESCRIPTION:
                       01AD   548 ;
                       01AD   549 ;       This routine returns a line from the input stream. It may be from the consol
                       01AD   550 ;       from a script file. If the first character of the user prompt from ASK$????
                       01AD   551 ;       <NULL> the operator is prompted directly without consulting the script.
                       01AD   552 ;       Syncronization between the scripts and DS_??LINE is accomplished by matching
                       01AD   553 ;       prompt string with the line in the script file.  If it does not match and
                       01AD   554 ;       DS_GETLINE was called the program is aborted, If DS_SUPERLINE was called the
                       01AD   555 ;       is ignored and the next line is examined. The prompt string and leading spac
                       01AD   556 ;       tabs are removed from the script line before it is returned to the program.
                       01AD   557 ;       addition lower case characters in the script are translated to uppercase. If
                       01AD   558 ;       input line is expected from the operator and the OPERTOR flag is clear a nul
                       01AD   559 ;       line is returned.  Currently the ASK$???? routines apply the program specifi
                       01AD   560 ;       default value in the case of a null line.
                       01AD   561 ;
                       01AD   562 ; CALLING SEQUENCE:
                       01AD   563 ;
                       01AD   564 ;       CALL    DS_GETLINE
                       01AD   565 ;       CALL    DS_SUPERLINE
                       01AD   566 ;
                       01AD   567 ; INPUT PARAMETERS:
                       01AD   568 ;
                       01AD   569 ;        4(AP) -        STRING BUFFER ADDRESS
                       01AD   570 ;        8(AP) -        LONGWORD FOR RETURNED STRING LENGTH
                       01AD   571 ;       12(AP) -        STRING BUFFER SIZE
                       01AD   572 ;       16(AP) -        Address of ASCIC user prompt
                       01AD   573 ;       20(AP) -        Address of ASCIC extra prompt text
```

```
01AD    575 ; IMPLICIT INPUTS:
01AD    576 ;
01AD    577 ;       NONE
01AD    578 ;
01AD    579 ; OUTPUT PARAMETERS:
01AD    580 ;
01AD    581 ;       NONE
01AD    582 ;
01AD    583 ; IMPLICIT OUTPUTS:
01AD    584 ;
01AD    585 ;       NONE
01AD    586 ;
01AD    587 ; COMPLETION CODES:
01AD    588 ;
01AD    589 ;       SS$_NORMAL
01AD    590 ;
01AD    591 ; REGISTER USAGE:
01AD    592 ;
01AD    593 ;       R4      Length of current line in script
01AD    594 ;       R5      Address of current line in script
01AD    595 ;       R6      Flags
01AD    596 ;                 bit 0 set if supervisor request, else clear
01AD    597 ;                 bit 1 set if already printed "prompt not found text"
01AD    598 ;       R7      Address of ASCIC user prompt
01AD    599 ;       R8      address of informational prompt
01AD    600 ;       R11     Base of current script block
01AD    601 ;
01AD    602 ; SIDE EFFECTS:
01AD    603 ;
01AD    604 ;       R1 = RETURNED STRING LENGTH
01AD    605 ;--
```

```
                        09FC  01AD   607          .ENTRY  DS_SUPERLINE,^M<R2,R3,R4,R5,R6,R7,R8,R11>
                              01AF   608
                 56     01    D0   01AF   609          MOVL    #1, R6                  ; Flag supervisor request
                        04    11   01B2   610          BRB     COMMON                  ; Join at common
                              01B4   611
                        09FC  01B4   612          .ENTRY  DS_GETLINE,^M<R2,R3,R4,R5,R6,R7,R8,R11>
                              01B6   613
                 56     D4    01B6   614          CLRL    R6                      ; Flag user request
                              01B8   615
                              01B8   616 COMMON:
           57 0000000B'EF     9E   01B8   617          MOVAB   L^T_NULL, R7            ; Initially a null user prompt          [07]
                 58     57    D0   01BF   618          MOVL    R7, R8                  ; Null information prompt
                 04     6C    D1   01C2   619          CMPL    (AP), #4                ; Is user prompt address present?
                        0D    1F   01C5   620          BLSSU   10$                     ; Branch if not
                 57     10 AC D0   01C7   621          MOVL    16(AP), R7              ; Yes use it
                 05     6C    D1   01CB   622          CMPL    (AP), #5                ; Is user info prompt present?
                        04    1F   01CE   623          BLSSU   10$                     ; Branch if not
                 58     14 AC D0   01D0   624          MOVL    20(AP), R8              ; Yes, use it
                              01D4   625
                              01D4   626 10$:     Br_If_Not_Script        20$     ; Branch if using console now           [12]
           5B 000000C0'EF     D0   01DC   627          MOVL    L*L_LINK, R11           ; Any script active?                    [07]
                        09    13   01E3   628          BEQL    20$                     ; Branch if no script active
                              01E5   629
                        67    95   01E5   630          TSTB    (R7)                    ; null user prompt?
                        08    13   01E7   631          BEQL    30$                     ; Yes, continue
                     01 A7    95   01E9   632          TSTB    1(R7)                   ; Is first character of prompt null?
                        03    12   01EC   633          BNEQ    30$                     ; Branch if can be scripted
                              01EE   634
                      0084    31   01EE   635 20$:     BRW     CONSOLE                 ; Branch to use console
                              01F1   636
                      0348    30   01F1   637 30$:     BSBW    ADVANCE                 ; Point to next record
                   0F 50      E8   01F4   638          BLBS    R0, 40$                 ; Branch if not <EOF>
           55 0000000C'EF     9E   01F7   639          MOVAB   L^T_EOF, R5             ; Text for "@ <EOF>"                    [07]
                 54     85    9A   01FE   640          MOVZBL  (R5)+, R4               ; Length of "@ <EOF>"
                      0080    30   0201   641          BSBW    PRINT_LINE              ; Type prompt and text--CH TO WORD DISP
                        B2    11   0204   642          BRB     COMMON                  ; And try again
```

```
                      54   D5  0206   644 40$:    TSTL    R4                          ; Any characters in line?
                      09   13  0208   645         BEQL    50$                         ; no, use it as is
                  65  21   91  020A   646         CMPB    #^A"!", (R5)                ; Comment line?
                      04   12  020D   647         BNEQ    50$                         ; No, use it
                      73   10  020F   648         BSBB    PRINT_LINE                  ; Type prompt and text
                      A5   11  0211   649         BRB     COMMON                      ; Try again
                           0213   650
                  53  57   D0  0213   651 50$:    MOVL    R7, R3                      ; Get address of user prompt
                  52  83   9A  0216   652         MOVZBL  (R3)+, R2                   ; Get length of ASCIC string
                     FDE4' 30  0219   653         BSBW    SCAN$COMPARE                ; Compare the strings
                      52   D5  021C   654         TSTL    R2                          ; Check for exhausted prompt string
                      4A   13  021E   655         BEQL    80$                         ; Branch if prompt match
                15 56  01   E2  0220   656         BBSS    #1, R6, 60$                 ; Branch if already told of error
                           0224   657
                      57   DD  C224   658         PUSHL   R7                          ; Address of prompt string
            00000047'EF    9F  0226   659         PUSHAB  L^T_PNF                     ; Edit string for "Prompt Not Found"    [07]
                  7E   01  9A  022C   660         movzbl  #DS$K_PrintF,-(SP)          ; Routine name constant                 [14]
                  7E   1E  98  022F   661         cvtbl   #DS$K_Type_Script_Pnf,-(SP)     ; The typecode constant             [14]
        00000000'GF    04  FB  0232   662         CALLS   #4, G^DSX$Print             ; Type it                               [14]
                           0239   663
                   22 56   E9  0239   664 60$:    BLBC    R6, 70$                     ; Branch if user request
                50   0C AB  3C  023C   665         MOVZWL  SC$W_CURR(R11), R0          ; Get current offset
                   10 AB40  9F  0240   666         PUSHAB  SC$A_BUF(R11)[R0]           ; Address of string
                7E   0A AB  3C  0244   667         MOVZWL  SC$W_LEN(R11), -(SP)        ; Length of string
            0000002A'EF    9F  0248   668         PUSHAB  L^T_SKIP                    ; Edit string for "skipping"            [07]
                  7E   01  9A  024E   669         movzbl  #DS$K_printf,-(sp)          ; Routine                               [14]
                  7E   1F  98  0251   670         cvtbl   #DS$K_type_script_skip,-(sp)    ; Print code.                       [14]
        00000000'GF    05  FB  0254   671         CALLS   #5, G^DSX$PRINT             ; Print it                              [14]
                     FF5A  31  0258   672         BRW     COMMON                      ; Branch to try again
                           025E   673
                0A AB  0A AB  B2  025E   674 70$:    MCOMW   SC$W_LEN(R11), -
                           0263   675                     SC$W_LEN(R11)               ; Complement len so record is used again
        00000000'9F    6C  FA  0263   676         CALLG   (AP), @#DS$ABORT            ; Program request...abort program
                           026A   677
                     FD93' 30  026A   678 80$:    BSBW    SCAN$SPACES                 ; Remove spaces and tabs from input
                           026D   679                                                 ;                                       [17]
              5C5C 8F    65  B1  026D   680         CMPW    (R5),#^X5C5C                ; CHECK FOR // AT THE END OF PROMPT     [17]
                      03   12  0272   681         BNEQ    82$                         ; IF NOT CONTINUE SCRIPT
                     002E  31  0274   682         BRW     CONSOLF                     ; IF YES PRINT PROMPT AND WAIT FOR ANSWER
                           0277   683 82$:                                            ;
                      0B   10  0277   684         BSBB    PRINT_LINE                  ; Type it
                     029C  30  0279   685         BSBW    COPY_IT                     ; Copy input to buffer
                           027C   686
                51   08 BC  D0  027C   687         MOVL    @8(AP), R1                  ; Get length of returned line
                   50   00' D0  0280   688         MOVL    S^#SS$_NORMAL, R0           ; It worked!
                           04  0283   689         RET                                 ; Return to caller
```

```
                                0284    691 ;+
                                0284    692 ; Little subroutine to print a line iff the Verify bit is set.
                                0284    693 ;-
                                0284    694
                                0284    695 PRINT_LINE:
                                0284    696         Br_If_Not_Verify        10$     ; Skip output if VERIFY is clear        [12]
                      7E    54  7D    028C    697         MOVQ    R4, -(SP)               ; Push length/address of data
                            57  DD    028F    698         PUSHL   R7                      ; Push address of ASCIC user prompt
            00000014'EF         9F    0291    699         PUSHAB  L^T_PROMPT              ; Type prompt and returned line         [07]
                      7E    01  9A    0297    700         movzbl  #DS$K_printf,-(sp)      ; routine.                              [14]
                      7E    20  98    029A    701         cvtbl   #DS$K_type_script_prompt,-(sp)  ; code                          [14]
          00000000'GF       06  FB    029D    702         CALLS   #6, G^DSX$PRINT         ; Type prompt and returned line         [14]
                                02A4    703
                                05  02A4    704 10$:    RSB                             ; Return                                [08]
```

```
                                02A5   706 ;+
                                02A5   707 ; No DS script currently active; get input from SYS$INPUT
                                02A5   708 ;-
                                02A5   709
                                02A5   710 CONSOLE:
          51   0000001D'EF   9E  02A5   711          MOVAB   L^T_PROMPT2, R1          ; Address of edit string                [07]
               50   81   9A  02AC   712          MOVZBL  (R1)+, R0                ; Get length of string
               7E   50   7D  02AF   713          MOVQ    R0, -(SP)                ; Push descriptor
           00000000'EF   9F  02B2   714          PUSHAB  L^DS$GT_BUFFER           ; Address of buffer                     [07]
            7E   0000'8F   3C  02B8   715          MOVZWL  #DS$K_BUFSIZ, -(SP)      ; Length of output buffer
                    52   D4  02BD   716          clrl    r2                       ; Indicator to save space               [09]
                         02BF   717          Br_If_Not_Binary        20$      ; If not BINARY, branch around          [12]
             50   04 AE   D0  02C7   718          movl    4(sp), r0                ; Get address of buffer                 [09]
                60   02   90  02CB   719          movb    #ds$k_type_user_prompt, -; Set first byte to the type code ...  [09]
                         02CE   720                   (r0)                     ; ... for general prompt                [09]
                   12 56   E9  02CE   721          blbc    r6, 10$                  ; Branch if it's not DS>                [09]
                67   04   91  02D1   722          cmpb    #4, (r7)                 ; Compare length to "DS> "              [09]
                     0D   12  02D4   723          bneq    10$                      ; If not, can't be DS>                  [09]
       203E5344 8F    01 A7  D1  02D6   724          cmpl    1(r7), #^A"DS> "         ; Is it actually DS>?                   [09]
                     03   12  02DE   725          bneq    10$                      ; Skip if not                           [09]
                60   01   90  02E0   726          movb    #ds$k_type_ds_prompt, - ; Otherwise, load ...                   [09]
                         02E3   727                   (r0)                     ; ... DS prompt code                    [09]
                         02E3   728
                     6E   D7  02E3   729 10$:     decl    (sp)                     ; Don't let FAO overwrite the type      [09]
                  04 AE   D6  02E5   730          incl    4(sp)                    ;                                       [09]
                     52   D6  02E8   731          incl    r2                       ; Remember that BINARY is set           [09]
                         02EA   732
                7E   57   7D  02EA   733 20$:     MOVQ    R7, -(SP)                ; Push user prompt, info prompt         [09]
                  08 AE   7F  02ED   734          PUSHAQ  8(SP)                    ; Address of buffer descriptor
                  0C AE   3F  02F0   735          PUSHAW  12(SP)                   ; Address to store final length
                  18 AE   7F  02F3   736          PUSHAQ  24(SP)                   ; Address of descriptor of edit string
       00000000'9F    05   FB  02F6   737          CALLS   #5, @#SYS$FAO            ; Edit prompt
                  54 52   E9  02FD   738          blbc    r2, 60$                  ; Branch if not BINARY                  [09]
                     6E   D6  0300   739          incl    (sp)                     ; Now include type code in string       [09]
                  04 AE   D7  0302   740          decl    4(sp)                    ;                                       [09]
```

```
                                0305   742 ;
                                0305   743 ; DEBUG ***
                                0305   744 ;
            48 00000000'EF   E9  0305   745          blbc    L^Debug_the_sucker, 50$ ; Skip debug stuff if flag clear        [99]
                        1F   BB  030C   746          pushr   #^M<r0, r1, r2, r3, r4> ; Make debug transparent                [99]
                     54  6E   9E  030E   747          movab   (sp), r4                ; Save current sp                       [99]
                        12   11  0311   748          brb     40$                     ; Skip format string                    [99]
                                0313   749
5A 21 3D 70 74 5B 0000031B'010E0000'  0313   750 30$:     .ascid  "[tp=!ZB]!/"            ; Format string                 [12]
                     2F 21 5D 42  0321
                                0325   751
                     5E  0A   C2  0325   752 40$:     subl2   #10, sp                 ; Allocate space on stack               [99]
                        6E   9F  0328   753          pushab  (sp)                    ; Set it's address                      [99]
                     7E  0A   9A  032A   754          movzbl  #10, -(sp)              ; And it's length                       [99]
                     53  6E   9E  032D   755          movab   (sp), r3                ; And remember it                       [99]
                  7E    18 B4   9A  0330   756          movzbl  @24(r4), -(sp)          ; Final argument for FAO                [99]
                        18 B4   94  0334   757          clrb    @24(r4)                 ; Clear it out                          [99]
                        63   9F  0337   758          pushab  (r3)                    ; Where to put length                   [99]
                        63   9F  0339   759          pushab  (r3)                    ; Address of result buffer              [99]
                     D5 AF   9F  033B   760          pushab  30$                     ; Address of format string              [99]
               00000000'GF   04  FB  033E   761          calls   #4, G^sys$fao           ; FAO it                                [99]
                        7E   63  7D  0345   762          movq    (r3), -(sp)             ; Push address of result string         [99]
               00000000'EF   02  FB  0348   763          calls   #2, dsx$type_out        ; Print it                              [99]
                     5E  64   9E  034F   764          movab   (r4), sp                ; Restore initial stack                 [99]
                        1F   BA  0352   765          popr    #^M<r0, r1, r2, r3, r4> ; Restore registers                     [99]
                                0354   766
                                0354   767 50$:                                     ;                                       [99]
                                0354   768
                                0354   769 ;
                                0354   770 ; DEBUG ***
                                0354   771 ;
                                0354   772
                     7E   7C  0354   773 60$:     CLRQ    -(SP)                   ; Space for IOSB                        [09]
                  21 56   E8  0356   774          BLBS    R6, 70$                 ; Branch if commmand mode
                              0359   775          Br_If_Oper      70$             ; If operator present, branch around    [12]
                     08 AE   7F  0361   776          PUSHAQ  8(SP)                   ; Address of prompt descriptor
            00000024'EF   9F  0364   777          PUSHAB  L^T_PROMPT3             ; Edit string address                   [07]
                  7E    01   9A  036A   778          movzbl  #DS$K_printf,-(sp)      ; Routine                               [14]
                  7E    20   98  036D   779          cvtbl   #DS$K_type_script_prompt,-(sp) ; Code.                          [14]
               00000000'GF   04  FB  0370   780          CALLS   #4, G^DSX$PRINT         ; Type it ...                           [14]
                        0192   31  0377   781          BRW     DS_GETLINE_OKAY_X       ; ... and exit w/null input line        [11]
```

```
                                037A   783 ;
                                037A   784 ; Separate flow into two primary cases: if DS$V_BATCH bit is set, then
                                037A   785 ; SYS$INPUT is a file, and we are in user mode: utilize RMS routines to
                                037A   786 ; manipulate file.  Otherwise, use QIO since it must run in standalone.
                                037A   787 ;
                                037A   788
                                037A   789 70$:    Br_If_Not_Batch 100$              ; Branch if terminal or mailbox control [12]
                                0382   790
                                0382   791 ;+
                                0382   792 ; RMS SYS$INPUT processing.  Use $GET to read record, and 'fake' echo
                                0382   793 ; via $PUT (PRINT call) to SYS$OUTPUT
                                0382   794 ;-
                                0382   795
      53   00000000'EF   9E    0382   796          MOVAB   DS$RAB_OUTPUT, R3         ; Point to output RAB
      52   00000000'EF   9E    0389   797          MOVAB   DS$RAB_INPUT, R2          ; Point to input RAB
        54    08 AE    9E    0390   798          MOVAB   8(SP), R4                 ; Save pointer to prompt descriptor
     24 A2    04 AC    D0    0394   799          MOVL    4(AP), RAB$L_UBF(R2)      ; Load input buffer address
     20 A2    0C AC    B0    0399   800          MOVW    12(AP), RAB$W_USZ(R2)     ;  And size of same
                             039E   801          $GET    (R2)                      ; Read a record
            11 50    E8    03A7   802          BLBS    R0, 90$                   ; Exit if error
0001827A 8F    50    D1    03AA   803          CMPL    R0, #RMS$_EOF             ; If this is end-of-file time
              05    12    03B1   804          BNEQ    80$                       ; :then
     50   0000'8F    3C    03B3   805          MOVZWL  #SS$_ENDOFFILE, R0        ; ::pretend it was SS end-of-file
                             03B8   806
            0154    31    03B8   807 80$:    BRW     DS_GETLINE_X              ; Exit routine                          [11]
                             03BB   808
        50   22 A2    3C    03BB   809 90$:    MOVZWL  RAB$W_RSZ(R2), R0         ; Get size of input line
           6E   01    B0    03BF   810          MOVW    #1, (SP)                  ; Set success code in IOSB
        02 AE   50    B0    03C2   811          MOVW    R0, 2(SP)                 ; Set it into IOSB for common code
             28 A2    DD    03C6   812          PUSHL   RAB$L_RBF(R2)             ; Address of input record
                50    DD    03C9   813          PUSHL   R0                        ; Size of input record
                54    DD    03CB   814          PUSHL   R4                        ; Pointer to prompt descriptor
          0000003E'EF    9F    03CD   815          PUSHAB  L^T_ECHO                  ; Pointer to ASCIC control string       [07]
               7E    01    9A    03D3   816          movzbl  #DS$K_printf,-(sp)        ; routine.                              [14]
               7E    21    98    03D6   817          cvtbl   #DS$K_type_script_echo,-(sp)    ; code                        [14]
     00000000'GF    06    FB    03D9   818          CALLS   #6, G^DSX$PRINT           ; Output                                [14]
              0109    31    03E0   819          BRW     CHECK_STATUS              ; Skip ahead                            [11]
```

```
                                03E3    821 ;+
                                03E3    822 ; QIO SYS$INPUT.  Attempt QIO read with prompt--this will fail if SYS$INPUT
                                03E3    823 ; is a mailbox.
                                03E3    824 ;-
                                03E3    825
                                03E3    826 100$:   Br_If_CRD_AutoTest_Off  110$    ; If running CRD-AutoTest, branch       [16]
                                03EB    827         Br_If_CRD_MenuTest_On   110$    ; If running CRD-MenuTest, branch       [16]
                                03F3    828         Br_If_CRD_MenuTest_Off  110$    ; If running CRD Menu Test, branch      [16]
                          45    11    03FB    829         Brb     114$                ; Else, skip all this CRD crap          [15]
                                03FD    830
                        21 56   E9    03FD    831 110$:   BlbC    R6, 112$            ; If user prompt, branch                [15]
                                0400    832
                                0400    833         ;+
                                0400    834         ; This is a DS prompt (i.e., "DS> "). Push the address and length of    [15]
                                0400    835         ; where to put the returned string as parameters 4 and 3.               [15]
                                0400    836         ;-
                                0400    837
                        04 BC   DF    0400    838         PushAL  @4 (AP)             ; Push address of string as param-4     [15]
                        0C AC   DD    0403    839         PushL   12 (AP)             ; Push length of string as param-3      [15]
                           01   DD    0406    840         PushL   #DS$K_Type_DS_Prompt    ; Push typecode as param-2          [15]
                           01   DD    0408    841         PushL   #CRD$R_TypeCode         ; Push type of call as param-1      [15]
                00000000'FF   04    FB    040A    842         CallS   #4, @L^DS$GA_CRD_DS_Interface   ; And call the routine!    [15]
                                0411    843
                                0411    844         ;+
                                0411    845         ; If the routine returns:                                               [15]
                                0411    846         ;       Success => return the string that this routine returned.        [15]
                                0411    847         ;                  (High word of R0 = length of the returned string).   [15]
                                0411    848         ;       Failure => do the QIO and return that string.                   [15]
                                0411    849         ; Note the the length of the returned string is stored in the high word [15]
                                0411    850         ; of R0. Thus, the Extract-Zero-Extended-Field instruction below.       [15]
                                0411    851         ;-
                                0411    852
                           2E 50    E9    0411    853         BlbC    R0, 114$            ; Do the QIO if the routine fails       [15]
                                0414    854
                51   50   10    10    EF    0414    855         ExtZV   #16, #16, R0, R1        ; Return the length of the string ...   [15]
                        08 BC   51    D0    0419    856         MovL    R1, @8 (AP)             ; ... in both R1 and the 2nd parameter  [15]
                           50   00'   D0    041D    857         MovL    S^#SS$_Normal, R0       ; Return success                        [15]
                                04    0420    858         Ret                             ; Return to the DS caller               [15]
                                0421    859
                                0421    860         ;+
                                0421    861         ; This is a user prompt. Push the address and length of the user's      [15]
                                0421    862         ; prompt as parameters 4 and 3.                                         [15]
                                0421    863         ;-
                                0421    864
                        0C AE   DD    0421    865 112$:   PushL   12 (SP)             ; Push address of prompt as param-4     [15]
                        08 AE   DD    0424    866         PushL   8  (SP)             ; Push length of prompt as param-3      [15]
                           02   DD    0427    867         PushL   #DS$K_Type_User_Prompt  ; Push typecode as param-2           [15]
                           01   DD    0429    868         PushL   #CRD$R_TypeCode         ; Push type of call as param-1      [15]
                00000000'FF   04    FB    042B    869         CallS   #4, @L^DS$GA_CRD_DS_Interface   ; And call the routine!    [15]
                                0432    870
                                0432    871         ;+
                                0432    872         ; If the routine returns:                                               [15]
                                0432    873         ;       Success => return the string that this routine returned.        [15]
                                0432    874         ;                  (High word of R0 = length of the returned string).   [15]
                                0432    875         ;       Failure => do the QIO and return that string.                   [15]
                                0432    876         ; Note the the length of the returned string is stored in the high word [15]
                                0432    877         ; of R0. Thus, the Extract-Zero-Extended-Field instruction below.       [15]
```

```
                                0432   878          ;-
                                0432   879
                          0D 50 E9     0432   880          BlbC    R0, 114$                ; Do the QIO if the routine fails       [15]
                                0435   881
          51    50    10    10 EF     0435   882          ExtZV   #16, #16, R0, R1        ; Return the length of the string ...   [15]
                08 BC   51    D0     043A   883          MovL    R1, @8 (AP)             ; ... in both R1 and the 2nd parameter  [15]
                   50   00'   D0     043E   884          MovL    S^#SS$_Normal, R0       ; Return success                        [15]
                                04     0441   885          Ret                             ; Return to the DS caller               [15]
                                0442   886
                   50   6E    7E     0442   887 114$:    MOVAQ   (SP), R0                ; Point to allocated IOSB               [15]
                                0445   888          Br_If_Not_User  115$            ; Skip next if not user mode            [12]
           000000C8'EF   01    88     044D   889          BISB    #1, DS$GB_QIOACT        ; Set input QIO active                  [15]
                                0454   890
                                0454   891 115$:    $QIOW_S     ,-                  ; No efn                                [07]
                                0454   892                     L^DS$GW_TTIN,-          ; Channel number
                                0454   893                     S^#IO$_READPROMPT,-     ; Function code
                                0454   894                     (R0),-                  ; IOSB
                                0454   895                     ,,-                     ; No ASTadr or ASTprm
                                0454   896                     P1=@4(AP),-             ; Buffer address
                                0454   897                     P2=12(AP),-             ; Buffer size
                                0454   898                     P5=12(R0),-             ; Prompt buffer address
                                0454   899                     P6=8(R0)                ; Prompt buffer size
                                0479   900
            000000C8'EF   94    0479   901 118$:    CLRB    DS$GB_QIOACT            ; Clear QIO active
          50    0000'8F   B1    047F   902          CMPW    #SS$_ILLIOFUNC, R0      ; Did readprompt fail?
                      03    13  0484   903          BEQL    120$                    ; Yes try w/ prompt
                    0063    31  0486   904          BRW     CHECK_STATUS            ; Branch to check the status            [11]
```

```
                                0489    906 ;+
                                0489    907 ; Mail box read: Fake "read with prompt" function via write followed by read.
                                0489    908 ;-
                                0489    909
                     50   6E    7E  0489    910 120$:   MOVAQ   (SP), R0                ; Current stack level
                                048C    911
                                048C    912         $QIOW_S     ,-                      ; No efn                                [07]
                                048C    913                 L^DS$GW_TTOUT,-         ; Channel number                        [11]
                                048C    914                 S^#IO$_WRITEVBLK,-      ; Function code                         [11]
                                048C    915                 (R0),-                  ; IOSB                                  [11]
                                048C    916                 P1=@12(R0),-            ; Prompt buffer address                 [11]
                                048C    917                 P2=8(R0)                ; Prompt buffer size                    [11]
                     5F 50    E9  04AD    918         BLBC    R0,DS_GETLINE_X         ; Finish if error on prompt             [11]
                     5C 6E    E9  04B0    919         BLBC    (SP),DS_GETLINE_X       ; Finish if error on prompt             [11]
                     50   6E    7E  04B3    920         MOVAQ   (SP),R0                 ; Current stack level
                                04B6    921         Br_If_Not_User  130$            ; Skip if not usermode                  [12]
          000000C8'EF    01    88  04BE    922         BISB    #1,DS$GB_QIOACT         ; Flag input QIO active
                                04C5    923
                                04C5    924 130$:   $QIOW_S     ,-                      ; No efn                                [07]
                                04C5    925                 L^DS$GW_TTIN,-          ; Channel number                        [11]
                                04C5    926                 S^#IO$_READVBLK,-       ; Function code                         [11]
                                04C5    927                 (R0),-                  ; IOSB                                  [11]
                                04C5    928                 P1=@4(AP),-             ; Buffer address                        [11]
                                04C5    929                 P2=12(AP)               ; Buffer size                           [11]
          000000C8'EF    94    04E6    930         CLRB    DS$GB_QIOACT            ; Flag input QIO inactive
                                04EC    931
                                04EC    932 ;+
                                04EC    933 ; Fall into the common completion part.                                         [11]
                                04EC    934 ;-
```

```
                                04EC   936 ;+
                                04EC   937 ; Common completion section, joined from RMS section and both QIO sections
                                04EC   938 ; via branch.  Check status and translate output.
                                04EC   939 ;-
                                04EC   940
                                04EC   941 CHECK_STATUS:                        ;                                       [11]
                    20 50   E9  04EC   942          BLBC    R0, DS_GETLINE_X    ; RETURN IF QIO FAILED                  [11]
                    1D 6E   E9  04EF   943          BLBC    (SP), DS_GETLINE_X  ; Return if QIO failed                  [11]
                                04F2   944
               50   0000'8F   3C  04F2   945          MOVZWL  #SS$_ENDOFFILE, R0  ; ASSUME END OF FILE
         04 AE   0001001A 8F   D1  04F7   946          CMPL    #^X1001A, 4(SP)     ; SINGLE CHARACTER TERM=^Z?
                       0E   13  04FF   947          BEQL    DS_GETLINE_X        ; Exit with EOF                     [11]
                                0501   948
                 54   02 AE   3C  0501   949          MOVZWL  2(SP), R4           ; GET STRING LENGTH
                 55   04 AC   D0  0505   950          MOVL    4(AP), R5           ; Address of input string
                        000C   30  0509   951          BSBW    COPY_IT             ; Copy and translate
                                050C   952
                                050C   953 ;+
                                050C   954 ; Here if no error
                                050C   955 ;-
                                050C   956
                                050C   957 DS_GETLINE_OKAY_X:                   ;                                       [11]
                    50   00'   D0  050C   958          MOVL    S^#SS$_NORMAL, R0   ; Return success!
                                050F   959
                                050F   960 DS_GETLINE_X:                        ;                                       [11]
                 51   02 AE   3C  050F   961          MOVZWL  2(SP), R1           ; Return the string length ...          [11]
                 08 BC   51   D0  0513   962          MOVL    R1, @8(AP)          ; ... in both places                    [11]
                              04  0517   963          RET                         ; Return to caller
```

```
                     0518   965          .SBTTL  COPY_IT -       Copy and translate
                 00000518   966          .PSECT  Code, Shr, Exe, NoWrt, Byte
                     0518   967 ;++
                     0518   968 ; FUNCTIONAL DESCRIPTION:
                     0518   969 ;
                     0518   970 ;       This routine is used to copy a line to the user buffer
                     0518   971 ;       It translates lower case alpha's to upper case.
                     0518   972 ;
                     0518   973 ; CALLING SEQUENCE:
                     0518   974 ;
                     0518   975 ;       BSBW    COPY_IT
                     0518   976 ;
                     0518   977 ; INPUT PARAMETERS:
                     0518   978 ;
                     0518   979 ;       NONE
                     0518   980 ;
                     0518   981 ; IMPLICIT INPUTS:
                     0518   982 ;
                     0518   983 ;       R4      Length of string to be copied
                     0518   984 ;       R5      Address of string to be copied
                     0518   985 ;       4(AP)   Address of user buffer
                     0518   986 ;       8(AP)   Address to store length of line returned
                     0518   987 ;       12(AP)  Length of user buf'er
                     0518   988 ;
                     0518   989 ; OUTPUT PARAMETERS:
                     0518   990 ;
                     0518   991 ;       NONE
                     0518   992 ;
                     0518   993 ; IMPLICIT OUTPUTS:
                     0518   994 ;
                     0518   995 ;       4(AP)   buffer written
                     0518   996 ;       @8(AP)  written with length of buffer
                     0518   997 ;
                     0518   998 ; SIDE EFFECTS:
                     0518   999 ;
                     0518  1000 ;       NONE
                     0518  1001 ;
                     0518  1002 ; COMPLETION CODES:
                     0518  1003 ;
                     0518  1004 ;       NONE
                     0518  1005 ;--
```

```
                        0518  1007 COPY_IT:
      0C AC    54   D1  0518  1008          CMPL    R4, 12(AP)              ; Is buffer large enough?
               04   15  051C  1009          BLEQ    10$                     ; Branch if it is
      54    0C AC   D0  051E  1010          MOVL    12(AP), R4              ; Shorten record
                        0522  1011 10$:
      08 BC    54   D0  0522  1012          MOVL    R4, @8(AP)              ; Store length of record
      50    04 AC   D0  0526  1013          MOVL    4(AP), R0               ; Get address of buffer
               0C   11  052A  1014          BRB     30$                     ; Start with count of length
         80    85   90  052C  1015 20$:     MOVB    (R5)+, (R0)+            ; Copy a byte
   04 FF A0    06   E1  052F  1016          BBC     #6, -1(R0), 30$         ; Branch if not alpha
      FF A0    20   8A  0534  1017          BICB2   #32, -1(R0)             ; force it to upper case
            F1 54   F4  0538  1018 30$:     SOBGEQ  R4, 20$                 ; Count and branch
                     05  053B  1019          RSB                             ; Return to caller
```

```
                                        .SBTTL  ADVANCE -       Advance to next record in buffer
                                        .PSECT  Code, Shr, Exe, NoWrt, Byte
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Simulate the RMS $GET routine, deblocking records from disk blocks.
                                ;       Note that .SC$W_CUR(R11) + SC$A_BUF(R11) is address of current record
                                ;       Also .SC$W_LEN(R11) is length of that record.
                                ;       If length is negative the same record is to be used again and length
                                ;       is the complement of the record length.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBB    ADVANCE
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       NONE
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       NONE
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       SC$W_CURR       Offset to current record
                                ;       SC$W_LEN        Length of current record
                                ;       R4              Length of current record
                                ;       R5              Address of current record
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       SS$_NORMAL
                                ;       SS$_ENDOFFILE
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       THE CURRENT SCRIPT DESCRIPTOR IS UPDATED.
                                ;
                                ; REGISTER USAGE:
                                ;
                                ;       R11     Assumed to contain the base for the current script block
                                ;       R5      Current offset within buffer
                                ;       R4      Length of current record
                                ;--
```

```
                          053C  1071 ADVANCE:
    55   0C AB   32       053C  1072          CVTWL   SC$W_CURR(R11), R5       ; Get current offset
    54   0A AB   32       0540  1073          CVTWL   SC$W_LEN(R11), R4        ; Get negated length
            12   19       0544  1074          BLSS    60$                      ; Branch if record should be used again
                          0546  1075
        55   54   C0       0546  1076          ADDL    R4, R5                   ; Skip previous record
    55   0E AB   B1       0549  1077          CMPW    SC$W_END(R11), R5        ; Past end of data?
            21   15       054D  1078          BLEQ    80$                      ; Branch if end of file
 54   10 AB45   9A       054F  1079          MOVZBL  SC$A_BUF(R11)[R5], R4    ; Get length of next record
            55   D6       0554  1080          INCL    R5                       ; Point to data
            03   11       0556  1081          BRB     70$                      ; Branch to finish
                          0558  1082
        54   54   D2       0558  1083 60$:     MCOML   R4, R4                   ; Fix length
                          055B  1084
    0C AB   55   B0       C55B  1085 70$:     MOVW    R5, SC$W_CURR(R11)       ; Put current offset back
    0C AB   55   B0       055F  1086          MOVW    R5, SC$W_CURR(R11)       ; Put current offset back
    0A AB   54   B0       0563  1087          MOVW    R4, SC$W_LEN(R11)        ; Set current length
 55   10 AB45   9E       0567  1088          MOVAB   SC$A_BUF(R11)[R5], R5    ; Absolute address of line
        50   00'  D0       056C  1089          MOVL    S^#SS$_NORMAL, R0        ; It worked!
                  05       056F  1090          RSB
                          0570  1091 80$:
          FC15   30       0570  1092          BSBW    SCRIPT$FINISH            ; Remove current script file
 50   0000'8F   3C       0573  1093          MOVZWL  #SS$_ENDOFFILE, R0       ; Indicate the end
                  05       0578  1094          RSB                              ; Return
                          0579  1095          .END
```

```
$$.TAB                          = 0000007C R   D   03
$$.TABEND                       = 000000C0 R   D   03
$$.TMP                          = 00000000     D
$$.TMP1                         = 00000001     D
$$.TMP2                         = 00000062
$$T1                            = 00000001     D
$ER                             = 00000001     D
$MODULE                           00000000 R   D   02
ADVANCE                           0000053C R   D   04
BEGIN                             ********     X   00
BIT...                          = 00000003     D
CHECK_STATUS                      000004EC R   D   04
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        00000444     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEST                        00000020     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
COMMON                            000001B8 R   D   04
CONSOLE                           000002A5 R   D   04
COPY_IT                           00000518 R   D   04
CRD$K_CRD_INITIALIZATION        = 00000000   G D
CRD$K_DS_CONTROL_C              = 00000001   G D
CRD$K_DS_FLAGS                  = 00000000   G D
CRD$K_FAILURE                   = 00000000     D
CRD$K_HOOK_POINT                = 00000000   G D
CRD$K_LAST_ACCESS_CODE          = 00000002   G D
CRD$K_LAST_DS_VARIABLE          = 00000002   G D
CRD$K_LAST_FUNCTION             = 00000002   G D
CRD$K_LAST_HOOK_POINT           = 00000001   G D
CRD$K_READ                      = 00000000   G D
CRD$K_SUCCESS                   = 00000001     D
CRD$K_TYPECODE                  = 00000001   G D
CRD$K_WRITE                     = 00000001   G D
DEBUG_THE_SUCKER                  ********     X   00
DS$ABORT                          ********     X   00
DS$CLI                            ********     X   00
DS$GA_CRD_DS_INTERFACE            ********     X   00
DS$GB_QIOACT                      000000C8 RG  D   03
DS$GL_CLIBASE                     ********     X   00
DS$GL_FLAGS                       ********     X   00
DS$GT_BUFFER                      ********     X   00
DS$GW_TTIN                        ********     X   00
DS$GW_TTOUT                       ********     X   00
DS$K_BUFSIZ                       ********     X   00
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
```

```
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$LOAD                           ********   X    00
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$RAB_INPUT                      ********   X    00
DS$RAB_OUTPUT                     ********   X    00
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMRON                     = 00000011     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INSFMEM                     = 00660050     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
```

```
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_BINARY                    = 00000001     D
DSA$V_CRD_AUTOTEST_OFF          = 0000000B     D
DSA$V_CRD_MENUTEST_OFF          = 00000010     D
DSA$V_CRD_MENUTEST_ON           = 00000012     D
DSA$V_OPER                      = 0000000C     D
DSA$V_QA                        = 0000000F     D
DSA$V_USER                      = 0000001C     D
DSA$V_VERIFY                    = 00000009     D
DSR$COMPLETION                    ********     X    00
DSX$PRINT                         ********     X    00
DSX$TYPE_OUT                      ********     X    00
DS_ERRSUP                         ********     X    00
DS_GETLINE                        000001B4 RG  D    04
DS_GETLINE_OKAY_X                 0000050C R   D    04
DS_GETLINE_X                      0000050F R   D    04
DS_SUPERLINE                      000001AD RG  D    04
EXE$ALONONPAGED                   ********     X    00
EXE$DEANONPAGED                   ********     X    00
FAB$B_DNS                       = 00000035     D
FAB$B_FNS                       = 00000034     D
FAB$C_BID                       = 00000003     D
FAB$C_BLN                       = 00000050     D
FAB$C_SEQ                       = 00000000     D
FAB$C_VAR                       = 00000002     D
FAB$L_ALQ                       = 00000010     D
FAB$L_DNA                       = 00000030     D
FAB$L_FNA                       = 0000002C     D
FAB$L_FOP                       = 00000004     D
FAB$V_CHAN_MODE                 = 00000002     D
FAB$V_FILE_MODE                 = 00000004     D
FAB$V_LNM_MODE                  = 00000000     D
FAB$W_GBC                       = 00000048     D
FALSE                           = 00000000     D
INIT_CONTEXT                      ********     X    00
IO$_READPROMPT                    ********     X    00
IO$_READVBLK                      ********     X    00
IO$_WRITEVBLK                     ********     X    00
L_FLAGS                           000000C4 R   D    03
L_LINK                            000000C0 R   D    03
OFF                             = 00000000     D
ON                              = 00000001     D
PRINT_LINE                        00000284 R   D    04
QIO$CLEANUP                       ********     X    00
RAB$B_RAC                       = 0000001E     D
RAB$C_BID                       = 00000001     D
RAB$C_BLN                       = 00000044     D
RAB$C_SEQ                       = 00000000     D
RAB$L_CTX                       = 00000018     D
RAB$L_RBF                       = 00000028     D
RAB$L_ROP                       = 00000004     D
RAB$L_UBF                       = 00000024     D
RAB$W_RSZ                       = 00000022     D
RAB$W_USZ                       = 00000020     D
RMS$_EOF                        = 0001827A     D
SC$A_BUF                          00000010     D
SC$C_BLN                        = 00000010     D
SC$L_LINK                         00000000     D
SC$W_CURR                         0000000C     D
SC$W_END                          0000000E     D
SC$W_LEN                          0000000A     D
SC$W_SIZE                         00000008     D
SCAN$COMPARE                      ********     X    00
SCAN$SPACES                       ********     X    00
SCRIPT$CONT                       00000175 RG  D    04
SCRIPT$FINISH                     00000188 R   D    04
SCRIPT$FLUSH                      00000151 RG  D    04
SCRIPT$INIT                       00000000 RG  D    04
SCRIPT$MINCORE                  = 00000400   G  D
SCRIPT$OPEN                       0000000A RG  D    04
SCRIPT$OPEN_X                     0000014B R   D    04
SCRIPT$STOP                       0000016C RG  D    04
SCRIPT_FAB                        0000002C R   D    03
SCRIPT_RAB                        0000007C R   D    03
SCRIPT_XAB                        00000000 R   D    03
SIZ...                          = 00000001     D
SS$_CONTROLC                      ********     X    00
SS$_ENDOFFILE                     ********     X    00
SS$_ILLIOFUNC                     ********     X    00
SS$_NORMAL                        ********     X    00
SYS$CLOSE                         ********   G      04
SYS$CONNECT                       ********   G      04
SYS$DISCONNECT                    ********   G      04
SYS$FAO                           ********     X    00
SYS$GET                           ********   G      04
SYS$HIBER                         ********     X    00
SYS$OPEN                          ********   G      04
SYS$QIO                           ********     X    00
SYS$QIOW                          ********   GX     00
SYS$READEF                        ********     X    00
SYS$WAKE                          ********     X    00
TRUE                            = 00000001     D
T_DEF_COM                         00000007 R   D    02
T_ECHO                            0000003E R   D    02
```

```
T_EOF                           0000000C R   D   02
T_NULL                          0000000B R   D   02
T_PNF                           00000047 R   D   02
T_PROMPT                        00000014 R   D   02
T_PROMPT2                       0000001D R   D   02
T_PROMPT3                       00000024 R   D   02
T_SKIP                          0000002A R   D   02
V_ECHOED                      = 00000001     D
XAB$C_FHC                     = 0000001D     D
XAB$C_FHCLEN                  = 0000002C     D
XAB$L_EBK                     = 00000010     D
XAB$L_NXT                     = 00000004     D
XAB$W_FFB                     = 00000014     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	0000006A (106.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
WORK	000000C9 (201.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000579 (1401.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$.IAB	=0000007C-R	198 (1)	196 (1) 197 (1) 198 (1)
$$.TABEND	=000000C0-R	198 (1)	196 (1) 197 (1) 198 (1)
$$.TMP	=00000000	198 (1)	197 (1) 198 (1)
$$.TMP1	=00000001	801 (1)	339 (1) 370 (1) 379 (1) 393 (1)
			395 (1) 801 (1)
$$.TMP2	=00000062	801 (1)	339 (1) 370 (1) 379 (1) 393 (1)
			395 (1) 801 (1)
$$T1	=00000C01	929 (1)	899 (1) 917 (1) 929 (1)
$ER	=00000001	165 (1)	
$MODULE	00000000-R	165 (1)	
ADVANCE	0000053C-R	1071 (1)	#-637 (1)
BEGIN	00000000-XR		134 (1) 423 (1)
BIT...	=00000003	158 (1)	149 (1) 150 (1) 157 (1) 158 (1)
CHECK_STATUS	000004EC-R	941 (1)	#-819 (1) #-904 (1)
CLI$K_BUFSIZ	=00000100	151 (1)	151 (1)
CLI$K_SIZE	00000444	151 (1)	
CLI$L_ADDRESS	00000018	151 (1)	
CLI$L_COMMAND	00000004	151 (1)	
CLI$L_DATA	0000001C	151 (1)	
CLI$L_FLAGS	00000000	151 (1)	
CLI$L_LAST	00000024	151 (1)	
CLI$L_NEXT	00000030	151 (1)	
CLI$L_PASS	0000002C	151 (1)	
CLI$L_SUBT	00000028	151 (1)	
CLI$L_TEST	00000020	151 (1)	
CLI$Q_BUFQWD	00000034	151 (1)	
CLI$Q_FILE	00000008	151 (1)	#-323 (1) #-325 (1)
CLI$Q_SECTION	00000010	151 (1)	
CLI$Q_TIME	0000043C	151 (1)	
CLI$T_BUFFER	0000003C	151 (1)	
CLI$V_ADAPTER	=00000018	151 (1)	
CLI$V_ADR	=0000000B	151 (1)	
CLI$V_ASCII	=00000013	151 (1)	
CLI$V_BREAK	=0000000A	151 (1)	
CLI$V_BRIEF	=0000001B	151 (1)	
CLI$V_BYTE	=0000000D	151 (1)	
CLI$V_CLEAR	=00000002	151 (1)	
CLI$V_DEC	=00000010	151 (1)	
CLI$V_DEFAULT	=0000000C	151 (1)	
CLI$V_DEPOSIT	=00000019	151 (1)	
CLI$V_EVENT	=00000008	151 (1)	
CLI$V_EXAM	=00000005	151 (1)	
CLI$V_FLAGS	=00000009	151 (1)	
CLI$V_HEX	=00000012	151 (1)	
CLI$V_KERNEL	=00000017	151 (1)	
CLI$V_LOAD	=00000006	151 (1)	
CLI$V_LONG	=0000000F	151 (1)	
CLI$V_NOTNUF	=00000001	151 (1)	
CLI$V_OCT	=00000011	151 (1)	
CLI$V_PREG	=0000001A	151 (1)	

```
CLI$V_QA                        =00000007      151     (1)
CLI$V_QACKLOOPLOOPS             =0000001C      151     (1)
CLI$V_QAERRORPRINTS             =0000001B      151     (1)
CLI$V_QAMULTIPLEPASS            =0000001F      151     (1)
CLI$V_QASUBTESTLOOPS            =0000001E      151     (1)
CLI$V_QATESTLOOPS               =0000001D      151     (1)
CLI$V_REG                       =00000014      151     (1)
CLI$V_REQUIRED                  =00000000      151     (1)
CLI$V_RUN                       =00000015      151     (1)
CLI$V_SET                       =00000003      151     (1)
CLI$V_SHOW                      =00000004      151     (1)
CLI$V_VALSEC                    =00000016      151     (1)
CLI$V_WORD                      =0000000E      151     (1)
COMMON                           000001B8-R    616     (1)   #-610     (1)   #-642     (1)   #-649     (1)   #-672     (1)
CONSOLE                          000002A5-R    710     (1)   #-635     (1)   #-682     (1)
COPY_IT                          00000518-R    1007    (1)   #-685     (1)   #-951     (1)
CRD$K_CRD_INITIALIZATION        =00000000      158     (1)
CRD$K_DS_CONTROL_C              =00000001      158     (1)
CRD$K_DS_FLAGS                  =00000000      158     (1)
CRD$K_FAILURE                   =00000000      158     (1)
CRD$K_HOOK_POINT                =00000000      158     (1)
CRD$K_LAST_ACCESS_CODE          =00000002      158     (1)
CRD$K_LAST_DS_VARIABLE          =00000002      158     (1)
CRD$K_LAST_FUNCTION             =00000002      158     (1)
CRD$K_LAST_HOOK_POINT           =00000001      158     (1)
CRD$K_READ                      =00000000      158     (1)
CRD$K_SUCCESS                   =00000001      158     (1)
CRD$K_TYPECODE                  =00000001      158     (1)   #-841     (1)   #-868     (1)
CRD$K_WRITE                     =00000001      158     (1)
DEBUG_THE_SUCKER                 00000000-XR                  137     (1)   #-745     (1)
DS$ABORT                         00000000-XR                  124     (1)     676     (1)
DS$CLI                           00000000-XR                  123     (1)
DS$GA_CRD_DS_INTERFACE           00000000-XR                  138     (1)     842     (1)     869     (1)
DS$GB_QIOACT                     000000C8-R    208     (1)   #-889     (1)   #-901     (1)   #-922     (1)   #-930     (1)
DS$GL_CLIBASE                    00000000-XR                  127     (1)     322     (1)
DS$GL_FLAGS                      00000000-XR                  131     (1)     361     (1)     362     (1)     420     (1)
                                                              468     (1)     477     (1)     485     (1)     626     (1)
                                                              789     (1)
DS$GT_BUFFER                     00000000-XR                  133     (1)     714     (1)
DS$GW_TTIN                       00000000-XR                  126     (1)   #-899     (1)   #-929     (1)
DS$GW_TTOUT                      00000000-XR                  126     (1)   #-917     (1)
DS$K_BUFSIZ                      00000000-XR                  133     (1)   #-715     (1)
DS$K_ERROR                      =00000002      149     (1)
DS$K_NORMAL                     =00000001      149     (1)
DS$K_PRINTB                     =00000002      157     (1)
DS$K_PRINTF                     =00000001      157     (1)   #-660     (1)   #-669     (1)   #-700     (1)   #-778     (1)
                                                              #-816     (1)
DS$K_PRINTI                     =00000000      157     (1)
DS$K_PRINTX                     =00000003      157     (1)
DS$K_SEVERE                     =00000004      149     (1)
DS$K_SUBSYS                     =00000066      149     (1)     149     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014      157     (1)
DS$K_TYPE_ABORT_TEST            =00000013      157     (1)
DS$K_TYPE_COMMAND_ERR           =00000015      157     (1)
DS$K_TYPE_COMMAND_OUT           =00000016      157     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A      157     (1)
DS$K_TYPE_DS_PROMPT             =00000001      157     (1)   #-726     (1)   #-840     (1)
```

```
DS$K_TYPE_DS_START              =0000001D       157     (1)
DS$K_TYPE_ERRDEV                =00000008       157     (1)
DS$K_TYPE_ERRHARD               =00000006       157     (1)
DS$K_TYPE_ERROR_BODY            =00000009       157     (1)
DS$K_TYPE_ERROR_END             =0000000A       157     (1)
DS$K_TYPE_ERRPREP               =0000001B       157     (1)
DS$K_TYPE_ERRSOFT               =00000007       157     (1)
DS$K_TYPE_ERRSUP                =00000004       157     (1)
DS$K_TYPE_ERRSYS                =00000005       157     (1)
DS$K_TYPE_ERR_HALT              =0000000D       157     (1)
DS$K_TYPE_EXCEPTION             =0000000C       157     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B       157     (1)
DS$K_TYPE_FIRST_PASS            =00000011       157     (1)
DS$K_TYPE_GENERAL               =00000000       157     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003       157     (1)
DS$K_TYPE_NO_TESTS              =00000012       157     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C       157     (1)
DS$K_TYPE_PROGRAM_END           =00000010       157     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017       157     (1)
DS$K_TYPE_PROGRAM_START         =0000000F       157     (1)
DS$K_TYPE_QIO_INVADP            =00000024       157     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022       157     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023       157     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021       157     (1)     #-817     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E       157     (1)     #-661     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020       157     (1)     #-701     (1)     #-779     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F       157     (1)     #-670     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019       157     (1)
DS$K_TYPE_START_ERR             =00000018       157     (1)
DS$K_TYPE_START_LIST            =00000025       157     (1)
DS$K_TYPE_SUMMARY               =0000000E       157     (1)
DS$K_TYPE_USER_PROMPT           =00000002       157     (1)     #-719     (1)     #-867     (1)
DS$K_WARNING                    =00000000       149     (1)
DS$LOAD                          00000000-XR                      123     (1)
DS$M_ABRTFLG                    =00000040       150     (1)
DS$M_BADTIME                    =00100000       150     (1)
DS$M_BATCH                      =00400000       150     (1)
DS$M_BRKCLR                     =00001000       150     (1)
DS$M_BRKPT                      =00000800       150     (1)
DS$M_CHARFLG                    =00000100       150     (1)
DS$M_CMDFLG                     =00000080       150     (1)
DS$M_CTRLC                      =00000001       150     (1)
DS$M_CTRLO                      =00010000       150     (1)
DS$M_DEVFLG                     =00000200       150     (1)
DS$M_DISABLCC                   =01000000       150     (1)
DS$M_DONFLG                     =00002000       150     (1)
DS$M_ERRFLG                     =00000010       150     (1)
DS$M_EXCEPT                     =00080000       150     (1)
DS$M_EXETST                     =00040000       150     (1)
DS$M_HLTFLG                     =00000008       150     (1)
DS$M_LODFLG                     =00000002       150     (1)
DS$M_MEMMGT                     =00008000       150     (1)
DS$M_OUTPUT                     =00800000       150     (1)
DS$M_RUBFLG                     =00000020       150     (1)
DS$M_SCRIPT                     =00200000       150     (1)
DS$M_SETIMR                     =02000000       150     (1)
DS$M_STRFLG                     =00000004       150     (1)
```

```
DS$M_SUBT                       =00004000      150     (1)
DS$M_SYSFLG                     =00000400      150     (1)
DS$M_TIMRON                     =00020000      150     (1)
DS$RAB_INPUT                     00000000-XR                    135     (1)     797     (1)
DS$RAB_OUTPUT                    00000000-XR                    135     (1)     796     (1)
DS$V_ABRTFLG                    =00000006      150     (1)
DS$V_BADTIME                    =00000014      150     (1)
DS$V_BATCH                      =00000016      150     (1)     #-789   (1)
DS$V_BRKCLR                     =0000000C      150     (1)
DS$V_BRKPT                      =0000000B      150     (1)
DS$V_CHARFLG                    =00000008      150     (1)
DS$V_CMDFLG                     =00000007      150     (1)
DS$V_CTRLC                      =00000000      150     (1)     #-362   (1)
DS$V_CTRLO                      =00000010      150     (1)
DS$V_DEVFLG                     =00000009      150     (1)
DS$V_DISABLCC                   =00000018      150     (1)
DS$V_DONFLG                     =0000000D      150     (1)
DS$V_ERRFLG                     =00000004      150     (1)
DS$V_EXCEPT                     =00000013      150     (1)
DS$V_EXETST                     =00000012      150     (1)
DS$V_HLTFLG                     =00000003      150     (1)
DS$V_LODFLG                     =00000001      150     (1)
DS$V_MEMMGT                     =0000000F      150     (1)
DS$V_OUTPUT                     =00000017      150     (1)
DS$V_RUBFLG                     =00000005      150     (1)
DS$V_SCRIPT                     =00000015      150     (1)     #-420   (1)     #-468   (1)     #-477   (1)     #-485   (1)
                                                               #-626   (1)
DS$V_SETIMR                     =00000019      150     (1)
DS$V_STRFLG                     =00000002      150     (1)     #-361   (1)
DS$V_SUBT                       =0000000E      150     (1)
DS$V_SYSFLG                     =0000000A      150     (1)
DS$V_TIMRON                     =00000011      150     (1)
DS$_ARITH                       =006600D0      149     (1)
DS$_ASBE                        =00660118      149     (1)
DS$_BADLINK                     =006600F0      149     (1)
DS$_BADTYPE                     =006600E8      149     (1)
DS$_BIIC                        =00660120      149     (1)
DS$_CHME                        =006600A8      149     (1)
DS$_CHMK                        =006600E0      149     (1)
DS$_DEVNAME                     =00660108      149     (1)
DS$_ERROR                       =00660002      149     (1)
DS$_FHWE                        =00660068      149     (1)
DS$_FRAGBUF                     =00660080      149     (1)
DS$_ICBUSY                      =006600C8      149     (1)
DS$_ICERR                       =006600C0      149     (1)
DS$_IHWE                        =00660060      149     (1)
DS$_ILLCHAR                     =00660018      149     (1)
DS$_ILLPAGCNT                   =00660078      149     (1)
DS$_ILLUNIT                     =00660100      149     (1)
DS$_INSFMEM                     =00660050      149     (1)
DS$_IPL2HI                      =006600B8      149     (1)
DS$_IVADDR                      =00660040      149     (1)
DS$_IVVECT                      =00660038      149     (1)
DS$_KRNLSTK                     =00660090      149     (1)
DS$_LOGIC                       =00660070      149     (1)
DS$_MCHK                        =00660088      149     (1)
DS$_MMOFF                       =00660058      149     (1)
```

```
DS$_NEEDUNIT                =006600F8      149     (1)
DS$_NODE                    =00660128      149     (1)
DS$_NOPCS                   =00660110      149     (1)
DS$_NORMAL                  =00660001      149     (1)
DS$_NOSUPPORT               =006600B1      149     (1)
DS$_NOTDON                  =00660030      149     (1)
DS$_NOTIMP                  =006600B0      149     (1)       149     (1)
DS$_NULLSTR                 =00660010      149     (1)
DS$_OVERFLOW                =00660008      149     (1)
DS$_POWER                   =00660098      149     (1)
DS$_PROGERR                 =00660020      149     (1)
DS$_SEVERE                  =00660004      149     (1)
DS$_TRANSL                  =006600A0      149     (1)
DS$_TRUNCATE                =00660028      149     (1)
DS$_UNEXPINT                =006600D8      149     (1)
DS$_VASFULL                 =00660048      149     (1)
DS$_WARNING                 =00660000      149     (1)       149     (1)
DSA$GL_FLAGS                 0000FE00                         310     (1)       466     (1)       696     (1)       717     (1)
                                                              775     (1)       826     (1)       827     (1)       828     (1)
                                                              888     (1)       921     (1)
DSA$V_BINARY                =00000001                      #-717     (1)
DSA$V_CRD_AUTOTEST_OFF      =0000000B                      #-826     (1)
DSA$V_CRD_MENUTEST_OFF      =00000010                      #-828     (1)
DSA$V_CRD_MENUTEST_ON       =00000012                      #-827     (1)
DSA$V_OPER                  =0000000C                      #-775     (1)
DSA$V_QA                    =0000000F                      #-466     (1)
DSA$V_USER                  =0000001C                      #-310     (1)     #-888     (1)     #-921     (1)
DSA$V_VERIFY                =00000009                      #-696     (1)
DSR$COMPLETION               00000000-XR                     124     (1)     #-340     (1)     #-399     (1)
DSX$PRINT                    00000000-XR                     123     (1)       662     (1)       671     (1)       702     (1)
                                                             780     (1)       818     (1)
DSX$TYPE_OUT                 00000000-XR                     136     (1)       763     (1)
DS_ERRSUP                    00000000-XR                     123     (1)
DS_GETLINE                   000001B4-R     612     (1)
DS_GETLINE_OKAY_X            0000050C-R     957     (1)    #-781     (1)
DS_GETLINE_X                 0000050F-R     960     (1)    #-807     (1)     #-918     (1)     #-919     (1)     #-942     (1)
                                                          #-943     (1)     #-947     (1)
DS_SUPERLINE                 000001AD-R     607     (1)
EXE$ALONONPAGED              00000000-XR                     132     (1)     #-351     (1)
EXE$DEANONPAGED              00000000-XR                     132     (1)     #-418     (1)     #-533     (1)
FAB$B_DNS                   =00000035                      #-327     (1)
FAB$B_FNS                   =00000034                      #-324     (1)
FAB$C_BID                   =00000003                       197     (1)
FAB$C_BLN                   =00000050                       197     (1)
FAB$C_SEQ                   =00000000                       197     (1)
FAB$C_VAR                   =00000002                       197     (1)
FAB$L_ALQ                   =00000010                       197     (1)
FAB$L_DNA                   =00000030                      #-329     (1)
FAB$L_FNA                   =0000002C                      #-326     (1)
FAB$L_FOP                   =00000004                       197     (1)
FAB$V_CHAN_MODE             =00000002                       197     (1)
FAB$V_FILE_MODE             =00000004                       197     (1)
FAB$V_LNM_MODE              =00000000                       197     (1)
FAB$W_GBC                   =00000048                       197     (1)
FALSE                       =00000000      158     (1)
INIT_CONTEXT                 00000000-XR                     134     (1)       309     (1)
IO$_READPROMPT               00000000-XR                     130     (1)     #-899     (1)
```

```
IO$_READVBLK                    00000000-XR                    130  (1)    #-929  (1)
IO$_WRITEVBLK                   00000000-XR                    130  (1)    #-917  (1)
L_FLAGS                         000000C4-R   203  (1)
L_LINK                          000000C0-R   200  (1)  #-264  (1)  #-359  (1)  #-360  (1)  #-470  (1)
                                                       #-483  (1)  #-530  (1)  #-532  (1)  #-535  (1)
                                                       #-627  (1)
OFF                             =00000000    158  (1)
ON                              =00000001    158  (1)
PRINT_LINE                       00000284-R  695  (1)  #-641  (1)  #-648  (1)  #-684  (1)
QIO$CLEANUP                     00000000-XR                    134  (1)    311  (1)
RAB$B_RAC                       =0000001E                      198  (1)
RAB$C_BID                       =00000001                      198  (1)
RAB$C_BLN                       =00000044                      198  (1)
RAB$C_SEQ                       =00000000                      198  (1)
RAB$L_CTX                       =00000018                      198  (1)
RAB$L_RBF                       =00000028                    #-812  (1)
RAB$L_ROP                       =00000004                      198  (1)
RAB$L_UBF                       =00000024                    #-373  (1)  #-799  (1)
RAB$W_RSZ                       =00000022                    #-381  (1)  #-809  (1)
RAB$W_USZ                       =00000020                    #-374  (1)  #-377  (1)  #-800  (1)
RMS$_EOF                        =0001827A                    #-397  (1)  #-803  (1)
SC$A_BUF                         00000010    222  (1)  #-1079  (1)    1088  (1)    363  (1)  #-364  (1)
                                                       #-410  (1)    666  (1)
SC$C_BLN                        =00000010    223  (1)    350  (1)
SC$L_LINK                        00000000    216  (1)  #-359  (1)  #-532  (1)
SC$W_CURR                        0000000C    220  (1)  #-1072  (1)  #-1085  (1)  #-1086  (1)  #-665  (1)
SC$W_END                         0000000E    221  (1)  #-1077  (1)  #-410  (1)
SC$W_LEN                         0000000A    219  (1)  #-1073  (1)  #-1087  (1)  #-667  (1)  #-674  (1)
                                                       #-675  (1)
SC$W_SIZE                        00000008    218  (1)  #-358  (1)  #-412  (1)
SCAN$COMPARE                    00000000-XR                    125  (1)  #-653  (1)
SCAN$SPACES                     00000000-XR                    125  (1)  #-678  (1)
SCRIPT$CONT                      00000175-R  481  (1)
SCRIPT$FINISH                    00000188-R  527  (1)  #-1092  (1)  #-400  (1)  #-469  (1)
SCRIPT$FLUSH                     00000151-R  465  (1)  #-308  (1)
SCRIPT$INIT                      00000000-R  263  (1)
SCRIPT$MINCORE                  =00000400    146  (1)
SCRIPT$OPEN                      0000000A-R  307  (1)
SCRIPT$OPEN_X                    0000014B-R  422  (1)  #-342  (1)  #-402  (1)
SCRIPT$STOP                      0000016C-R  476  (1)  #-265  (1)  #-482  (1)  #-538  (1)
SCRIPT_FAB                       0000002C-R  197  (1)    198  (1)    321  (1)
SCRIPT_RAB                       0000007C-R  198  (1)    334  (1)
SCRIPT_XAB                       00000000-R  196  (1)    197  (1)    317  (1)
SIZ...                          =00000001    150  (1)    150  (1)
SS$_CONTROLC                    00000000-XR                    129  (1)
SS$_ENDOFFILE                   00000000-XR                  #-1093  (1)    129  (1)  #-805  (1)  #-945  (1)
SS$_ILLIOFUNC                   00000000-XR                    129  (1)  #-902  (1)
SS$_NORMAL                      00000000-XR                  #-1089  (1)    129  (1)  #-540  (1)  #-688  (1)
                                                       #-857  (1)  #-884  (1)  #-958  (1)
SYS$CLOSE                       00000000-XR                    395  (1)
SYS$CONNECT                     00000000-XR                    370  (1)
SYS$DISCONNECT                  00000000-XR                    393  (1)
SYS$FAO                         00000000-XR                    128  (1)    737  (1)    761  (1)
SYS$GET                         00000000-XR                    379  (1)    801  (1)
SYS$HIBER                       00000000-XR                    128  (1)
SYS$OPEN                        00000000-XR                    339  (1)
SYS$QIO                         00000000-XR                    128  (1)
```

```
SYS$QIOW        00000000-XR              128   (1)   899   (1)   917   (1)   929   (1)
SYS$READEF      00000000-XR              128   (1)
SYS$WAKE        00000000-XR              128   (1)
TRUE            =00000001     158  (1)
T_DEF_COM       00000007-R    167  (1)   328   (1)
T_ECHO          0000003E-R    188  (1)   815   (1)
T_EOF           0000000C-R    173  (1)   639   (1)
T_NULL          0000000B-R    170  (1)   617   (1)
T_PNF           00000047-R    191  (1)   659   (1)
T_PROMPT        00000014-R    176  (1)   699   (1)
T_PROMPT2       0000001D-R    179  (1)   711   (1)
T_PROMPT3       00000024-R    182  (1)   777   (1)
T_SKIP          0000002A-R    185  (1)   668   (1)
V_ECHOED        =00000001     206  (1)
XAB$C_FHC       =0000001D                196   (1)
XAB$C_FHCLEN    =0000002C                196   (1)
XAB$L_EBK       =00000010               #-348  (1)
XAB$L_NXT       =00000004                196   (1)
XAB$W_FFB       =00000014               #-349  (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$$R_SET	1	196 (1)	196 (1)
$$R_TABINIT	1	196 (1)	196 (1) 197 (1) 198 (1)
$$R_VBFSET	1	197 (1)	197 (1) 198 (1)
$$R_XSET	1	196 (1)	196 (1)
$CLOSE	1	395 (1)	395 (1)
$CONNECT	1	370 (1)	370 (1)
$CRD_LITERALS	2	158 (1)	158 (1)
$DEF	1	158 (1)	
$DEFINI	1	148 (1)	148 (1) 152 (1) 153 (1) 154 (1) 155 (1) 156 (1)
$DISCONNECT	1	393 (1)	393 (1)
$DS_DSADEF	5	148 (1)	148 (1)
$DS_DSDEF	2	149 (1)	149 (1)
$DS_TYPEDEF	4	157 (1)	157 (1)
$EQU	1	158 (1)	149 (1) 157 (1) 158 (1)
$EQULS1	1	158 (1)	149 (1) 157 (1) 158 (1)
$EQULST	1	149 (1)	149 (1) 157 (1) 158 (1)
$FAB	4	197 (1)	197 (1)
$FABDEF	1	152 (1)	152 (1) 197 (1)
$GBLINI	2		149 (1) 150 (1) 157 (1) 158 (1)
$GET	1	379 (1)	379 (1) 801 (1)
$OPEN	1	339 (1)	339 (1)
$PUSHADR	1	899 (1)	899 (1) 917 (1) 929 (1)
$PUSHTWO	1	899 (1)	899 (1) 917 (1) 929 (1)
$QIOPUSH	1	899 (1)	899 (1) 917 (1) 929 (1)
$QIOW_S	1	891 (1)	891 (1) 912 (1) 924 (1)
$RAB	2	198 (1)	198 (1)
$RABDEF	1	154 (1)	154 (1) 198 (1)
$RMSCALL	2	339 (1)	339 (1) 370 (1) 379 (1) 393 (1) 395 (1) 801 (1)
$RMSDEF	11	153 (1)	153 (1)
$VIELD	1		150 (1)
$VIELDI	1	58 (1)	150 (1)
$XABDEF	1	155 (1)	155 (1) 196 (1)
$XABFHC	1	196 (1)	196 (1)
$XABFHCDEF	1	156 (1)	156 (1) 196 (1)
BR_IF_CRD_AUTOTEST_OFF	1	826 (1)	826 (1)
BR_IF_CRD_MENUTEST_OFF	1	828 (1)	828 (1)
BR_IF_CRD_MENUTEST_ON	1	827 (1)	827 (1)
BR_IF_NOT_BATCH	1	789 (1)	789 (1)
BR_IF_NOT_BINARY	1	717 (1)	717 (1)
BR_IF_NOT_SCRIPT	1	626 (1)	626 (1)
BR_IF_NOT_USER	1	888 (1)	888 (1) 921 (1)
BR_IF_NOT_VERIFY	1	696 (1)	696 (1)
BR_IF_OPER	1	775 (1)	775 (1)
BR_IF_SCRIPT	1	468 (1)	468 (1)
BR_IF_USER	1	310 (1)	310 (1)
CLEAR_CTRLC	1	362 (1)	362 (1)
CLEAR_QA	1	466 (1)	466 (1)
CLEAR_SCRIPT	1	477 (1)	477 (1)

CLEAR_STRFLG	1	361	(1)	361	(1)		
CLIDEF	3	151	(1)	151	(1)		
DSFDEF	3	150	(1)	150	(1)		
MODNAM	1	165	(1)	165	(1)		
SET_SCRIPT	1	420	(1)	420	(1)	485	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.11	00:00:00.25
Command processing	145	00:00:00.89	00:00:02.48
Pass 1	1065	00:00:17.13	00:00:29.85
Symbol table sort	0	00:00:01.30	00:00:02.71
Pass 2	324	00:00:03.76	00:00:05.02
Symbol table output	46	00:00:00.28	00:00:00.31
Psect synopsis output	10	00:00:00.04	00:00:00.04
Cross-reference output	153	00:00:01.62	00:00:02.31
Assembler run totals	1778	00:00:25.13	00:00:42.97

The working set limit was 1000 pages.
92599 bytes (181 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 927 non-local and 56 local symbols.
1095 source lines were read in Pass 1, producing 0 object records in Pass 2.
134 pages of virtual memory were used to define 54 macros.

Macro library statistics

Macro library name	Macros defined
DRB1:[DS.WORK]DIAG.MLB;955	3
DRB1:[DS.WORK]DS.MLB;218	19
DRB1:[DS.WORK]CRD.MLB;12	1
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	27
TOTALS (all libraries)	50

1326 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) SCRIPT/UPDA=(SCRIPT.UPD,SCRIPT.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

```
;    0001  0     %Title '*** SHOWMEM Handle Show Memory command'
;    0002  0     Module ShowMemory (                          ! Supervisor SHOW MEMORY command
;    0003  0                     Ident = '06-06',
;    0004  0                     Addressing_Mode (External = Long_Relative),
;    0005  0                     Debug,
;    0006  0                     OptLevel = 3,
;    0007  0                     NoZip
;    0008  0                     ) =
;    0009  0     !
;    0010  0     ! Copyright (c) 1982, 1983, 1984
;    0011  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0012  0     !
;    0013  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0014  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0015  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0016  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0017  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0018  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0019  0     ! REMAIN IN DEC.
;    0020  0     !
;    0021  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0022  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0023  0     ! CORPORATION.
;    0024  0     !
;    0025  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0026  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0027  0     !
;    0028  0     !
;    0029  0     !++
;    0030  0     ! Facility:
;    0031  0     !
;    0032  0     !       Diagnostic Supervisor
;    0033  0     !
;    0034  0     ! Abstract:
;    0035  0     !
;    0036  0     !       List physical memory map (virtual in user mode).
;    0037  0     !
;    0038  0     ! Environment:
;    0039  0     !
;    0040  0     !       Diagnostic Supervisor
;    0041  0     !
;    0042  0     ! Author:
;    0043  0     !
;    0044  0     !       Dave Butenhof           05-Feb-1982
;    0045  0     !
;    0046  0     ! Modified by:
;    0047  0     !
;    0048  0     ! [01] Dave Butenhof, 25-Mar-1982, version 6.7
;    0049  0     !       Fix order of non-paged pool and lookaside list typeouts,
;    0050  0     !       mask out region select bits (31, 32) on P0BR and P1BR addresses
;    0051  0     !       (to make them physical, not virtual), and change top-of-DS and
;    0052  0     !       physical-memory-size messages to show Kb, not bytes (easier to
;    0053  0     !       read).  Also, print amount of available buffer space on /Buffer,
;    0054  0     !       and cut down on code some by setting all option bits if /All is
;    0055  0     !       set, instead of checking it throughout.
;    0056  0     !
;    0057  0     !  [02] Richard Brown, 9-July-82, Version 6.8
```

```
;    0058  0      !       Add printing of debugger memory allocation.
;    0059  0      !
;    0060  0      !  [03] Jack Stansbury, 14-September-1982, Version 6.9
;    0061  0      !       Added showing the buffer count.
;    0062  0      !
;    0063  0      !       04      Bob Bergazzi    May 17, 1983    version 6.11
;    0064  0      !               Changed the order of searching libraries.
;    0065  0      !
;    0066  0      !       05      Bob Bergazzi    Aug 29,1983     Version 6.13
;    0067  0      !               Added the VRSETMEM routine which sets a new memory
;    0068  0      !               size and then starts VDS at 10000. Doesn't get
;    0069  0      !               here in user mode.
;    0070  0      !
;    0071  0      !       06      Bob Bergazzi    Feb 14, 1984    Version 6.14
;    0072  0      !               Fixed VRSETMEM to call INI$LOAD_DEVICE which
;    0073  0      !               makes ptables for the console and boot devices.
;    0074  0      !--
;    0075  0
;    0076  1      begin
;    0077  1      !
;    0078  1      ! Table of contents:
;    0079  1      !
;    0080  1
;    0081  1      !
;    0082  1      ! Include files:
;    0083  1      !
;    0084  1
;    0085  1      Library '$Diag';                                                    !                                       [04]
;    0086  1
;    0087  1      Library '$Ds';                                                      !                                       [04]
;    0088  1
;    0089  1      Library 'Sys$Library:Lib';                                          !                                       [04]
;    0090  1
;    0091  1      !
;    0092  1      ! Macros:
;    0093  1      !
;    0094  1
;    0095  1
;    0096  1      !
;    0097  1      ! Psect Definitions:
;    0098  1      !
;    0099  1
;    0100  1      Psect Plit   = Data (NoExecute, Share, NoWrite, Addressing_Mode (Long_Relative));
;    0101  1      Psect Own    = Work (NoExecute, NoShare, Write, Addressing_Mode (Long_Relative));
;    0102  1      Psect Global = Work (NoExecute, NoShare, Write, Addressing_Mode (Long_Relative));
;    0103  1      Psect Code   = Code (Execute, Share, NoWrite);
;    0104  1
;    0105  1      !
;    0106  1      ! Equated Symbols:
;    0107  1      !
;    0108  1
;    0109  1      $Ds_DsaDef;                                                                 ! Define DSA locations
;    0110  1      $Ds_TypeDef;                                                                ! Define type codes
;    0111  1
;    0112  1      Literal                                                                     ! Bits in ShowMemoryFlags
;    0113  1          Sm_V_Map = 0,                                                           ! /Map
;    0114  1          Sm_V_Buffer = 1,                                                        ! /Buffer
```

```
;    0115  1          Sm_V_Data = 2,                                  ! /Data_Structure
;    0116  1          Sm_V_All = 3,                                   ! /All
;    0117  1          CR = 13,
;    0118  1          LF = 10;
;    0119  1
;    0120  1      Bind                                                ! Format strings
;    0121  1          Format = $Ascic ('!22<!XL !-(%D!SL)!> : !AC!/'),
;    0122  1          Format2= $Ascic ('!22<!XL !-(%D!SL)!> : !AC!AC!/'),
;    0123  1          Format3= $Ascic ('!22<!XL !-(%D!SL)!>  : !AC!AC!/'),
;    0124  1          Format4= $Ascic ('!22<!XL (!SLKb)!> : !AC!/'),                                          [01]
;    0125  1          Format5= $Ascic ('!22<!XL (!SLKb)!> : !AC!AC!/'),                                       [01]
;    0126  1          Format6= $Ascic ('!22  : !AC!/');                                  [02]
;    0127  1
;    0128  1      Bind                                                ! Common text parts
;    0129  1          T_Reserved  = $Ascic ('Reserved page'),
;    0130  1          T_DS        = $Ascic ('Diagnostic Supervisor'),
;    0131  1          T_Kernel    = $Ascic ('Kernel'),
;    0132  1          T_Interrupt = $Ascic ('Interrupt'),
;    0133  1          T_Executive = $Ascic ('Executive'),
;    0134  1          T_Supervisor= $Ascic ('Supervisor'),
;    0135  1          T_User      = $Ascic ('User'),
;    0136  1          T_Stack     = $Ascic (' stack base'),
;    0137  1          T_SP        = $Ascic (' SP'),
;    0138  1          T_P0        = $Ascic ('P0'),
;    0139  1          T_P1        = $Ascic ('P1'),
;    0140  1          T_System    = $Ascic ('System'),
;    0141  1          T_Base      = $Ascic (' Page Table base'),
;    0142  1          T_Length    = $Ascic (' Page Table length (longwords)'),
;    0143  1          T_P0_BufCnt = $Ascic ('Pages in P0 space in use for buffers'),      !                   [03]
;    0144  1          T_P1_BufCnt = $Ascic ('Pages in P1 space in use for buffers'),      !                   [03]
;    0145  1          T_S0_BufCnt = $Ascic ('Pages in System space in use for buffers'),  !                   [03]
;    0146  1          T_BufferSpace = $Ascic ('Available  buffer space ');                !                   [01]
;    0147  1
;    0148  1      !
;    0149  1      ! Own storage:
;    0150  1      !
;    0151  1
;    0152  1      Global
;    0153  1          ShowMemoryFlags : BitVector [4];
;    0154  1
;    0155  1      !
;    0156  1      ! External references:
;    0157  1      !
;    0158  1
;    0159  1      External
;    0160  1          Ds$GB_TypeCode : Byte,                          ! Type code
;    0161  1          Ds$GL_Flags,                                    ! Common flags (LODFLG)
;    0162  1          DS$GL_BufCnt : VECTOR [3, LONG],                ! The three buffer counts           [03]
;    0163  1          Ax_Buff,                                        ! Actual end of DS image
;    0164  1          Phd_Base,                                       ! Base of Process Header
;    0165  1          Pcb_Base,                                       ! Base of Hardware PCB
;    0166  1          Ds$AX_Jib,                                      ! Base of pseudo JIB
;    0167  1          Ds$AX_Arb,                                      ! Base of pseudo ARB
;    0168  1          Ds$AX_SoftPCB,                                  ! Base of pseudo software PCB
;    0169  1          Scb_Base,                                       ! Base of SCB
;    0170  1          Stack_Base,                                     ! Base of stack space
;    0171  1          KStkPtr,                                        ! Beginning of Kernel stack
```

```
;    0172  1          IStkPtr,                                  !
;    0173  1          EStkPtr,                                  !
;    0174  1          SStkPtr,                                  !
;    0175  1          UStkPtr,                                  !
;    0176  1          Ds$A_PrgBgn,                              ! Where program starts
;    0177  1          POPT_Base,                                ! End of built-in storage (_LAST)
;    0178  1          Ds$GL_LookAside,                          ! Start of Look-aside IRP list
;    0179  1          Exe$GL_Splitadr,                          ! Address of non-paged pool split
;    0180  1          Ds$GA_LastAdr,                            ! End of run-time DS
;    0181  1          Ds$GL_MemSize,                            ! Size of physical memory
;    0182  1          DS$GL_SET_MEMSIZE,                        ! Size of physical memory desired      [05]
;    0183  1          debug_lo,                                 ! Bottom of debugger                   [02]
;    0184  1          debug_hi;                                 ! Top of debugger                      [02]
;    0185  1
;    0186  1      Forward Routine
;    0187  1          DSR$CheckLoad  : Jsb_None,
;    0188  1          DSV$ShowMemory : Jsb_None NoValue,
;    0189  1          ShowMemory     : NoValue,
;    0190  1          VRSETMEM       : JSB_CLI ;               !                                      [05]
```

```
;    0191  1     %Sbttl 'Is a program loaded, really?'
;    0192  1
;    0193  1     Global Routine DSR$CheckLoad : Jsb_None =                                                    !C03
;    0194  2     Begin                                                                                        !C03
;    0195  2         Return ((.Ds$GL_Flags <Ds$V_LodFlg, 0>) And (.L$L_Environ <Env$V_Super, Env$S_Super> Eql Env$_Super));  !C03
;    0196  1     End;                                                                                         !C03


                                                        .TITLE  SHOWMEMORY *** SHOWMEM Handle Show Memory comma
;                                                                       nd
                                                        .IDENT  \06-06\

                                                        .PSECT  WORK,NOEXE,2

                                                00000 SHOWMEMORYFLAGS::
                                                        .BLKB   1

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

21 44 25 28 2D 21 20 4C 58 21 3C 32 32 21 1B 00000 P.AAA:   .ASCII  <27>\!22<!XL !-(%D!SL)!> : !AC!/\          ;
      2F 21 43 41 21 20 3A 20 3E 21 29 4C 53 0000F                                                          ;
21 44 25 28 2D 21 20 4C 58 21 3C 32 32 21 1E 0001C P.AAB:   .ASCII  <30>\!22<!XL !-(%D!SL)!> : !AC!AC!/\       ;
21 43 41 21 43 41 21 20 3A 20 3E 21 29 4C 53 0002B                                                          ;
                           2F 0003A                                                                         ;
21 44 25 28 2D 21 20 4C 58 21 3C 32 32 21 1F 0003B P.AAC:   .ASCII  <31>\!22<!XL !-(%D!SL)!> :  !AC!AC!/\      ;
43 41 21 43 41 21 20 20 3A 20 3E 21 29 4C 53 0004A                                                          ;
                              2F 21 00059                                                                   ;
62 4B 4C 53 21 28 20 4C 58 21 3C 32 32 21 19 0005B P.AAD:   .ASCII  <25>\!22<!XL (!SLKb)!> : !AC!/\          ;
      2F 21 43 41 21 43 41 21 20 3A 20 3E 21 29 0006A                                                       ;
62 4B 4C 53 21 28 20 4C 58 21 3C 32 32 21 1C 00075 P.AAE:   .ASCII  <28>\!22<!XL (!SLKb)!> : !AC!AC!/\       ;
   2F 21 43 41 21 43 41 21 20 3A 20 3E 21 29 00084                                                          ;
      2F 21 43 41 21 20 3A 20 20 32 32 21 0C 00092 P.AAF:   .ASCII  <12>\!22  : !AC!/\                        ;
   65 67 61 70 20 64 65 76 72 65 73 65 52 0D 0009F P.AAG:   .ASCII  <13>\Reserved page\                       ;
70 75 53 20 63 69 74 73 6F 6E 67 61 69 44 15 000AD P.AAH:   .ASCII  <21>\Diagnostic Supervisor\              ;
                        72 6F 73 69 76 72 65 000BC                                                          ;
                        6C 65 6E 72 65 4B 06 000C3 P.AAI:   .ASCII  <6>\Kernel\                           ;
                  74 70 75 72 72 65 74 6E 49 09 000CA P.AAJ:   .ASCII  <9>\Interrupt\                       ;
                  65 76 69 74 75 63 65 78 45 09 000D4 P.AAK:   .ASCII  <9>\Executive\                       ;
               72 6F 73 69 76 72 65 70 75 53 0A 000DE P.AAL:   .ASCII  <10>\Supervisor\                     ;
                                 72 65 73 55 04 000E9 P.AAM:   .ASCII  <4>\User\                           ;
            65 73 61 62 20 6B 63 61 74 73 20 0B 000EE P.AAN:   .ASCII  <11>\ stack base\                    ;
                                    50 53 20 03 000FA P.AAO:   .ASCII  <3>\ SP\                           ;
                                    30 50 02 000FE P.AAP:   .ASCII  <2>\P0\                               ;
                                    31 50 02 00101 P.AAQ:   .ASCII  <2>\P1\                               ;
                        6D 65 74 73 79 53 06 00104 P.AAR:   .ASCII  <6>\System\                           ;
61 62 20 65 6C 62 61 54 20 65 67 61 50 20 10 0010B P.AAS:   .ASCII  <16>\ Page Table base\                  ;
                                          65 73 0011A                                                       ;
65 6C 20 65 6C 62 61 54 20 65 67 61 50 20 1E 0011C P.AAT:   .ASCII  <30>\ Page Table length (longwords)\    ;
73 64 72 6F 77 67 6E 6F 6C 28 20 68 74 67 6E 0012B                                                          ;
                                       29 0013A                                                             ;
70 73 20 30 50 20 6E 69 20 73 65 67 61 50 24 0013B P.AAU:   .ASCII  \$Pages in P0 space in use for buffers\  ;
20 72 6F 66 20 65 73 75 20 6E 69 20 65 63 61 0014A                                                          ;
                        73 72 65 66 66 75 62 00159                                                          ;
70 73 20 31 50 20 6E 69 20 73 65 67 61 50 24 00160 P.AAV:   .ASCII  \$Pages in P1 space in use for buffers\  ;
20 72 6F 66 20 65 73 75 20 6E 69 20 65 63 61 0016F                                                          ;
                        73 72 65 66 66 75 62 0017E                                                          ;
65 74 73 79 53 20 6E 69 20 73 65 67 61 50 28 00185 P.AAW:   .ASCII  \(Pages in System space in use for buffer\  ;
```

```
20 65 73 75 20 6E 69 20 65 63 61 70 73 20 6D 00194
                72 65 66 66 75 62 20 72 6F 66 001A3
                                        73 001AD          .ASCII  \s\                                    ;
66 75 62 20 20 65 6C 62 61 6C 69 61 76 41 18 001AE P.AAX:  .ASCII  <24>\Available  buffer space \        ;
                20 65 63 61 70 73 20 72 65 66 001BD                                                      ;
```

```
DSA$AL_APTMAIL=     65024
DSA$GL_FLAGS=       65024
DSA$GL_APTCOM=      65028
DSA$GL_PASSES=      65032
DSA$GL_UNITS=       65036
DSA$GL_SECTNO=      65040
DSA$GL_SID=         65044
DSA$GL_MSGTYP=      65088
DSA$GL_ERRNO=       65092
DSA$GL_EVENT=       65096
DSA$GL_SUBTNO=      65100
DSA$GL_TESTNO=      65104
DSA$GL_PASSNO=      65108
DSA$GL_DEVLEN=      65112
DSA$GL_DEVNAM=      65116
DSA$GL_MSGPTR=      65128
FORMAT=             P.AAA
FORMAT2=            P.AAB
FORMAT3=            P.AAC
FORMAT4=            P.AAD
FORMAT5=            P.AAE
FORMAT6=            P.AAF
T_RESERVED=         P.AAG
T_DS=               P.AAH
T_KERNEL=           P.AAI
T_INTERRUPT=        P.AAJ
T_EXECUTIVE=        P.AAK
T_SUPERVISOR=       P.AAL
T_USER=             P.AAM
T_STACK=            P.AAN
T_SP=               P.AAO
T_P0=               P.AAP
T_P1=               P.AAQ
T_SYSTEM=           P.AAR
T_BASE=             P.AAS
T_LENGTH=           P.AAT
T_P0_BUFCNT=        P.AAU
T_P1_BUFCNT=        P.AAV
T_S0_BUFCNT=        P.AAW
T_BUFFERSPACE=      P.AAX
        .EXTRN  DS$GB_TYPECODE, DS$GL_FLAGS
        .EXTRN  DS$GL_BUFCNT, AX_BUFF
        .EXTRN  PHD_BASE, PCB_BASE
        .EXTRN  DS$AX_JIB, DS$AX_ARB
        .EXTRN  DS$AX_SOFTPCB, SCB_BASE
        .EXTRN  STACK_BASE, KSTKPTR
        .EXTRN  ISTKPTR, ESTKPTR
        .EXTRN  SSTKPTR, USTKPTR
        .EXTRN  DS$A_PRGBGN, POPT_BASE
        .EXTRN  DS$GL_LOOKASIDE
        .EXTRN  EXE$GL_SPLITADR
```

```
                                                                .EXTRN  DS$GA_LASTADR, DS$GL_MEMSIZE
                                                                .EXTRN  DS$GL_SET_MEMSIZE
                                                                .EXTRN  DEBUG_LO, DEBUG_HI

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                50  D4 00000 DSR$CHECKLOAD::
                                                             CLRL    R0                                          ; 0195
          01 00000204    9F              08     02  ED 00002         CMPZV   #2, #8, @#^X00000204, #1            ;
                                                02  12 0000B         BNEQ    1$                                  ;
                                                50  D6 0000D         INCL    R0                                  ;
          51 00000000G   EF              01     01  EF 0000F 1$:     EXTZV   #1, #1, DS$GL_FLAGS, R1             ;
                                                51  51  D2 00018     MCOML   R1, R1                              ;
                                                50  51  CA 0001B     BICL2   R1, R0                              ;
                                                    05  0001E        RSB                                         ; 0196

; Routine Size:  31 bytes,    Routine Base:  CODE + 0000
```

```
;    0197  1     %Sbttl 'DSV$ShowMemory routine'
;    0198  1
;    0199  1     Global Routine DSV$ShowMemory : Jsb_None NoValue =
;    0200  2         Begin
;    0201  2         ShowMemory ()                                   ! Just CALLS routine
;    0202  1         End;


                        0000V  CF           00  FB 00000 DSV$SHOWMEMORY::
                                                         CALLS   #0, SHOWMEMORY                  ; 0201
                                            05 00005     RSB                                     ; 0202

; Routine Size:  6 bytes,    Routine Base:  CODE + 001F
```

```
:    0203  1     %Sbttl 'ShowMemory Routine'
:    0204  1
:    0205  1     Routine ShowMemory : NoValue =
:    0206  2         Begin
:    0207  2
:    0208  2         Local
:    0209  2             Temp,
:    0210  2             MemoryFlags : BitVector [4],
:    0211  2             OnLine : Byte;
:    0212  2
:    0213  2         Builtin
:    0214  2             Mfpr;
:    0215  2
:    0216  2         MemoryFlags = .ShowMemoryFlags<0, 4>;              ! Get flags locally
:    0217  2
:    0218  2         If .MemoryFlags Eql 0                              ! Default is /Map
:    0219  2         Then
:    0220  2             MemoryFlags [Sm_V_Map] = 1;
:    0221  2
:    0222  2         If .MemoryFlags [Sm_V_All]                         ! Set all option bits if /All                    [01]
:    0223  2         Then                                               !                                                [01]
:    0224  3             Begin                                          !                                                [01]
:    0225  3             MemoryFlags [Sm_V_Map] = 1;                    !                                                [01]
:    0226  3             MemoryFlags [Sm_V_Buffer] = 1;                 !                                                [01]
:    0227  3             MemoryFlags [Sm_V_Data] = 1                    !                                                [01]
:    0228  2             End;                                           !                                                [01]
:    0229  2
:    0230  2         OnLine = .Dsa$V_User;                              ! Get user mode bit locally
:    0231  2         Ds$GB_TypeCode = Ds$K_Type_Command_Out;
:    0232  2
:    0233  2         If .MemoryFlags [Sm_V_Map]                         !                                                [01]
:    0234  2         Then
: P  0235  2             $Ds_PrintF (                                   ! First page of memory
: P  0236  2                 Format,
: P  0237  2                 0,
: P  0238  2                 T_Reserved
:    0239  2                 );
:    0240  2
:    0241  2         If .MemoryFlags [Sm_V_Map]                         !                                                [01]
:    0242  2         Then
: P  0243  2             $Ds_PrintF (                                   ! Begin of diagnostic
: P  0244  2                 Format,
: P  0245  2                 Ds$A_PrgBgn,
: P  0246  2                 $Ascic ('Beginning of diagnostic region')
:    0247  2                 );
:    0248  2
:    0249  2         If .MemoryFlags [Sm_V_Map] And DSR$CheckLoad ()    !                                                [01]
:    0250  2         Then
: P  0251  2             $Ds_PrintF (                                   ! End of diagnostic
: P  0252  2                 Format,
: P  0253  2                 .L$A_LastAdr,
: P  0254  2                 $Ascic ('End of loaded diagnostic')
:    0255  2                 );
:    0256  2
:    0257  2         If .MemoryFlags [Sm_V_Buffer]                      ! If /Buffer or /All                             [01]
:    0258  2          And Dsr$CheckLoad ()                              ! .. and diagnostic is loaded                    [01]
:    0259  2         Then                                               !                                                [01]
```

```
;  P 0260  2          $Ds_PrintF (                                ! Available buffer space                    [01]
;  P 0261  2              Format5,                                !                                           [01]
;  P 0262  2              %X'FA00' - .L$A_LastAdr,                ! .. amount available                       [01]
;  P 0263  2              (%X'FA00' - .L$A_LastAdr)/1024,         ! .. same in Kb                             [01]
;  P 0264  2              T_BufferSpace,                          !                                           [01]
;  P 0265  2              $Ascic ('below Supervisor')             !                                           [01]
;    0266  2              );                                      !                                           [01]
;    0267  2
;    0268  2          If .MemoryFlags [Sm_V_Map] Or .MemoryFlags [Sm_V_Data]      !                           [01]
;    0269  2          Then
;  P 0270  2              $Ds_PrintF (                            ! APT Ptext buffer
;  P 0271  2                  Format,
;  P 0272  2                  %X'FA00',
;  P 0273  2                  $Ascic ('APT Ptext buffer')
;    0274  2                  );
;    0275  2
;    0276  2          If .MemoryFlags [Sm_V_Map]                  !                                           [01]
;    0277  2           Or .MemoryFlags [Sm_V_Data]
;    0278  2          Then
;  P 0279  2              $Ds_PrintF (                            ! APT mailbox
;  P 0280  2                  Format,
;  P 0281  2                  %X'FE00',
;  P 0282  2                  $Ascic ('APT Mailbox')
;    0283  2                  );
;    0284  2
;    0285  2          If .MemoryFlags [Sm_V_Map]                  !                                           [01]
;    0286  2          Then
;  P 0287  2              $Ds_PrintF (                            ! Begin of Supervisor
;  P 0288  2                  Format2,
;  P 0289  2                  %X'10000',
;  P 0290  2                  $Ascic ('Beginning of '),
;  P 0291  2                  T_DS
;    0292  2                  );
;    0293  2
;    0294  2          If .MemoryFlags [Sm_V_Map]                  !                                           [01]
;    0295  2          Then
;  P 0296  2              $Ds_PrintF (                            ! Top of DS image
;  P 0297  2                  Format,
;  P 0298  2                  Ax_Buff,
;  P 0299  2                  $Ascic ('End of Supervisor load image')
;    0300  2                  );
;    0301  2
;    0302  2          If .MemoryFlags [Sm_V_Data] And Not .OnLine  !                                          [01]
;    0303  2          Then
;    0304  3              Begin
;  P 0305  3              $Ds_PrintF (                            ! PHD
;  P 0306  3                  Format,
;  P 0307  3                  Phd_Base,
;  P 0308  3                  $Ascic ('PHD')
;    0309  3                  );
;  P 0310  3              $Ds_PrintF (                            ! hardware PCB
;  P 0311  3                  Format,
;  P 0312  3                  Pcb_Base,
;  P 0313  3                  $Ascic ('Hardware PCB')
;    0314  3                  );
;  P 0315  3              $Ds_PrintF (                            ! JIB
;  P 0316  3                  Format,
```

```
; P 0317  3                   Ds$AX_Jib,
; P 0318  3                   $Ascic ('JIB')
;    0319  3                   );
; P 0320  3               $Ds_PrintF (                        ! ARB
; P 0321  3                   Format,
; P 0322  3                   Ds$AX_Arb,
; P 0323  3                   $Ascic ('ARB')
;    0324  3                   );
; P 0325  3               $Ds_PrintF (                        ! software PCB
; P 0326  3                   Format,
; P 0327  3                   Ds$AX_SoftPCB,
; P 0328  3                   $Ascic ('Software PCB')
;    0329  3                   );
; P 0330  3               $Ds_PrintF (                        ! SCB
; P 0331  3                   Format,
; P 0332  3                   SCB_Base,
; P 0333  3                   $Ascic ('SCB')
;    0334  3                   );
; P 0335  3               $Ds_PrintF (                        ! Reserved page
; P 0336  3                   Format,
; P 0337  3                   Stack_Base,
; P 0338  3                   T_Reserved
;    0339  3                   );
;    0340  3               Mfpr (Pr$_Ksp, Temp);
; P 0341  3               $Ds_PrintF (                        ! Kernel Stack pointer
; P 0342  3                   Format2,                        !                          [03]
; P 0343  3                   .Temp,
; P 0344  3                   T_Kernel,
; P 0345  3                   T_SP
;    0346  3                   );
; P 0347  3               $Ds_PrintF (                        ! Kernel Stack pointer base
; P 0348  3                   Format3,                        !                          [03]
; P 0349  3                   KStkPtr,
; P 0350  3                   T_Kernel,
; P 0351  3                   T_Stack
;    0352  3                   );
;    0353  3               Mfpr (Pr$_Isp, Temp);
; P 0354  3               $Ds_PrintF (                        ! Interrupt Stack pointer
; P 0355  3                   Format2,                        !                          [03]
; P 0356  3                   .Temp,
; P 0357  3                   T_Interrupt,
; P 0358  3                   T_SP
;    0359  3                   );
; P 0360  3               $Ds_PrintF (                        ! Interrupt Stack pointer base
; P 0361  3                   Format3,                        !                          [03]
; P 0362  3                   IStkPtr,
; P 0363  3                   T_Interrupt,
; P 0364  3                   T_Stack
;    0365  3                   );
;    0366  3               Mfpr (Pr$_Esp, Temp);
; P 0367  3               $Ds_PrintF (                        ! Executive Stack pointer
; P 0368  3                   Format2,                        !                          [03]
; P 0369  3                   .Temp,
; P 0370  3                   T_Executive,
; P 0371  3                   T_SP
;    0372  3                   );
; P 0373  3               $Ds_PrintF (                        ! Executive Stack pointer base
```

```
;  P 0374  3                Format3,                        !                                                       [03]
;  P 0375  3                EStkPtr,
;  P 0376  3                T_Executive,
;  P 0377  3                T_Stack
;    0373  3                );
;    0379  3            Mfpr (Pr$_Ssp, Temp);
;  P 0380  3            $Ds_PrintF (                        ! Supervisor Stack pointer
;  P 0381  3                Format2,                        !                                                       [03]
;  P 0382  3                .Temp,
;  P 0383  3                T_Supervisor,
;  P 0384  3                T_SP
;    0385  3                );
;  P 0386  3            $Ds_PrintF (                        ! Supervisor Stack pointer base
;  P 0387  3                Format3,                        !                                                       [03]
;  P 0388  3                SStkPtr,
;  P 0389  3                T_Supervisor,
;  P 0390  3                T_Stack
;    0391  3                );
;    0392  3            Mfpr (Pr$_Usp, Temp);
;  P 0393  3            $Ds_PrintF (                        ! User Stack pointer
;  P 0394  3                Format2,                        !                                                       [03]
;  P 0395  3                .Temp,
;  P 0396  3                T_User,
;  P 0397  3                T_SP
;    0398  3                );
;  P 0399  3            $Ds_PrintF (                        ! User Stack pointer base
;  P 0400  3                Format3,                        !                                                       [03]
;  P 0401  3                UStkPtr,
;  P 0402  3                T_User,
;  P 0403  3                T_Stack
;    0404  3                );
;    0405  3            Mfpr (Pr$_P0BR, Temp);
;  P 0406  3            $Ds_PrintF (                        ! P0 Page table
;  P 0407  3                Format2,
;  P 0408  3                .Temp<0, 30>,                   !                                                       [01]
;  P 0409  3                T_P0,
;  P 0410  3                T_Base
;    0411  3                );
;    0412  3            Mfpr (Pr$_P0LR, Temp);
;  P 0413  3            $Ds_PrintF (                        ! P0 Page table
;  P 0414  3                Format3,
;  P 0415  3                .Temp,
;  P 0416  3                T_P0,
;  P 0417  3                T_Length
;    0418  3                );
;    0419  3            Mfpr (Pr$_P1BR, Temp);
;  P 0420  3            $Ds_PrintF (                        ! P1 Page table
;  P 0421  3                Format2,
;  P 0422  3                .Temp<0, 30>,                   !                                                       [01]
;  P 0423  3                T_P1,
;  P 0424  3                T_Base
;    0425  3                );
;    0426  3            Mfpr (Pr$_P1LR, Temp);
;  P 0427  3            $Ds_PrintF (                        ! P1 Page table
;  P 0428  3                Format3,
;  P 0429  3                .Temp,
;  P 0430  3                T_P1,
```

```
;P 0431  3                  T_Length
;  0432  3                  );
;  0433  3              Mfpr (Pr$_SBR, Temp);
;P 0434  3              $Ds_PrintF (                                ! System Page table
;P 0435  3                  Format2,
;P 0436  3                  .Temp,
;P 0437  3                  T_System,
;P 0438  3                  T_Base
;  0439  3                  );
;  0440  3              Mfpr (Pr$_SLR, Temp);
;P 0441  3              $Ds_PrintF (                                ! System Page table
;P 0442  3                  Format3,
;P 0443  3                  .Temp,
;P 0444  3                  T_System,
;P 0445  3                  T_Length
;  0446  4                  )
;  0447  2              End;
;  0448  2
;  0449  2          If .MemoryFlags [Sm_V_Map] Or .MemoryFlags [Sm_V_Data]      !                                          [01]
;  0450  2          Then
;  0451  3              Begin
;P 0452  3              $Ds_PrintF (                                ! Non-Paged Pool
;P 0453  3                  Format,
;P 0454  3                  .Ds$GL_LookAside,
;P 0455  3                  $Ascic ('Start of Supervisor memory pool')
;  0456  3                  );
;  0457  3
;  0458  3              If .Ds$GL_LookAside Neq .Exe$GL_SplitAdr
;  0459  3              Then
;P 0460  3                  $Ds_PrintF (                            ! Lookaside list start
;P 0461  3                      Format,
;P 0462  3                      .Exe$GL_SplitAdr,
;P 0463  3                      $Ascic ('Start of IRP Look-aside list')
;  0464  3                      );
;  0465  3
;  0466  2              End;
;  0467  2
;  0468  2          If .MemoryFlags [Sm_V_Map]                      !                                          [01]
;  0469  2           Or .MemoryFlags [Sm_V_Data]
;  0470  2          Then
;P 0471  2              $Ds_PrintF (                                ! Top of run-time DS
;P 0472  2                  Format5,
;P 0473  2                  .Ds$GA_LastAdr,
;P 0474  2                  (.Ds$GA_LastAdr + 1023)/1024,           ! # of Kb (next higher)                    [01]
;P 0475  2                  $Ascic ('Top of run-time '),
;P 0476  2                  T_DS
;  0477  2                  );
;  0478  2
;  0479  2          If .MemoryFlags [Sm_V_Buffer]                   ! If /Buffer or /All                       [01]
;  0480  3           And (.Ds$GL_MemSize Gtr .Ds$GA_LastAdr)        !                                          [01]
;  0481  2          Then                                            !                                          [01]
;P 0482  2              $Ds_PrintF (                                ! Available buffer space                   [01]
;P 0483  2                  Format5,                                !                                          [01]
;P 0484  2                  .Ds$GL_MemSize - .Ds$GA_LastAdr,        ! .. amount available                      [01]
;P 0485  2                  (.Ds$GL_MemSize - .Ds$GA_LastAdr)/1024, ! .. same in Kb                            [01]
;P 0486  2                  T_BufferSpace,                          !                                          [01]
;P 0487  2                  $Ascic ('above Supervisor')             !                                          [01]
```

```
;   0488  2                 );                                          !                                                   [01]
;   0489  2
;   0490  2             If .MemoryFlags [Sm_V_Buffer] Then              ! If /Buffer or /All                                [03]
;   0491  3                 Begin                                       !                                                   [03]
; P 0492  3                 $DS_PrintF (                                ! Print the amount of buffer space being used in    [03]
; P 0493  3                     Format,                                 ! . . . P0 space.                                   [03]
; P 0494  3                     .DS$GL_BufCnt [0],                      ! . . . This is the number of pages                 [03]
; P 0495  3                     T_P0_BufCnt                             !                                                   [03]
;   0496  3                     );                                      !                                                   [03]
; P 0497  3                 $DS_PrintF (                                ! Print the amount of buffer space being used in    [03]
; P 0498  3                     Format,                                 ! . . . P1 space.                                   [03]
; P 0499  3                     .)S$GL_BufCnt [1],                      ! . . . This is the number of pages                 [03]
; P 0500  3                     T_P1_BufCnt                             !                                                   [03]
;   0501  3                     );                                      !                                                   [03]
; P 0502  3                 $DS_PrintF (                                ! Print the amount of buffer space being used in    [03]
; P 0503  3                     Format,                                 ! . . . S0 space.                                   [03]
; P 0504  3                     .DS$GL_BufCnt [2],                      ! . . . This is the number of pages                 [03]
; P 0505  3                     T_S0_BufCnt                             !                                                   [03]
;   0506  3                     );                                      !                                                   [03]
;   0507  2                 End;                                        !                                                   [03]
;   0508  2
;   0509  2             If .dsa$v_debug                                 !If debugger is loaded, print its limits.           [02]
;   0510  2             then
;   0511  3                 begin
; P 0512  3                 $ds_printf (format,
; P 0513  3                             .debug_lo,
;   0514  3                             $ascic ('Bottom of debugger'));
;   0515  3                 if not .online
;   0516  3                 then
; P 0517  3                     $ds_printf (format6,
;   0518  3                                 $ascic (' (includes allocation for symbol tables)'));
; P 0519  3                 $ds_printf (format,
; P 0520  3                             .debug_hi,
;   0521  3                             $ascic ('Top of debugger'));
;   0522  2                 end;
;   0523  2
;   0524  2             If Not .OnLine
;   0525  2             Then
; P 0526  2                 $Ds_PrintF (                                ! Top of physical memory
; P 0527  2                     Format4,
; P 0528  2                     .Ds$GL_MemSize,
; P 0529  2                     .Ds$GL_MemSize/1024,                    ! Kb of memory                                      [01]
; P 0530  2                     $Ascic ('Top of physical memory')
;   0531  2                     );
;   0532  2             Ds$GB_TypeCode = 0;                             ! Clear sticky typecode
;   0533  1             End;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

64 20 66 6F 20 67 6E 69 6E 6E 69 67 65 42 1E 001C7 P.AAY:   .ASCII  <30>\Beginning of diagnostic region\      ;
6F 69 67 65 72 20 63 69 74 73 6F 6E 67 61 69 001D6                                                          ;
                                          6E 001E5                                                          ;
20 64 65 64 61 6F 6C 20 66 6F 20 64 6E 45 18 001E6 P.AAZ:   .ASCII  <24>\End of loaded diagnostic\          ;
                  63 69 74 73 6F 6E 67 61 69 64 001F5                                                       ;
73 69 76 72 65 70 75 53 20 77 6F 6C 65 62 10 001FF P.ABA:   .ASCII  <16>\below Supervisor\                  ;
```

```
                                                        72  6F  0020E                                                         ;
66  66  75  62  20  74  78  65  74  50  20  54  50  41  10  00210 P.ABB:   .ASCII  <16>\APT Ptext buffer\                       ;
                                                        72  65  0021F                                                         ;
                78  6F  62  6C  69  61  4D  20  54  50  41  0B  00221 P.ABC:   .ASCII  <11>\APT Mailbox\                            ;
    20  66  6F  20  67  6E  69  6E  6E  69  67  65  42  0D  0022D P.ABD:   .ASCII  <13>\Beginning of \                          ;
69  76  72  65  70  75  53  20  66  6F  20  64  6E  45  1C  0023B P.ABE:   .ASCII  <28>\End of Supervisor load image\           ;
    65  67  61  6D  69  20  64  61  6F  6C  20  72  6F  73  0024A                                                         ;
                                            44  48  50  03  00258 P.ABF:   .ASCII  <3>\PHD\                                     ;
            42  43  50  20  65  72  61  77  64  72  61  48  0C  0025C P.ABG:   .ASCII  <12>\Hardware PCB\                           ;
                                            42  49  4A  03  00269 P.ABH:   .ASCII  <3>\JIB\                                     ;
                                            42  52  41  03  0026D P.ABI:   .ASCII  <3>\ARB\                                     ;
            42  43  50  20  65  72  61  77  74  66  6F  53  0C  00271 P.ABJ:   .ASCII  <12>\Software PCB\                           ;
                                            42  43  53  03  0027E P.ABK:   .ASCII  <3>\SCB\                                     ;
72  65  70  75  53  20  66  6F  20  74  72  61  74  53  1F  00282 P.ABL:   .ASCII  <31>\Start of Supervisor memory pool\        ;
6F  70  20  79  72  6F  6D  65  6D  20  72  6F  73  69  76  00291                                                         ;
                                                        6C  6F  002A0                                                         ;
4C  20  50  52  49  20  66  6F  20  74  72  61  74  53  1C  002A2 P.ABM:   .ASCII  <28>\Start of IRP Look-aside list\           ;
    74  73  69  6C  20  65  64  69  73  61  2D  6B  6F  6F  002B1                                                         ;
6D  69  74  2D  6E  75  72  20  66  6F  20  70  6F  54  10  002BF P.ABN:   .ASCII  <16>\Top of run-time \                       ;
                                                        20  65  002CE                                                         ;
73  69  76  72  65  70  75  53  20  65  76  6F  62  61  10  002D0 P.ABO:   .ASCII  <16>\above Supervisor\                       ;
                                                        72  6F  002DF                                                         ;
75  62  65  64  20  66  6F  20  6D  6F  74  74  6F  42  12  002E1 P.ABP:   .ASCII  <18>\Bottom of debugger\                     ;
                                            72  65  67  67  002F0                                                         ;
6C  6C  61  20  73  65  64  75  6C  63  6E  69  28  20  28  002F4 P.ABQ:   .ASCII  \( (includes allocation for symbol tables\ ;
6D  79  73  20  72  6F  66  20  6E  6F  69  74  61  63  6F  00303                                                         ;
                        73  65  6C  62  61  74  20  6C  6F  62  00312                                                         ;
                                                            29  0031C          .ASCII  \)\                                  ;
65  67  67  75  62  65  64  20  66  6F  20  70  6F  54  0F  0031D P.ABR:   .ASCII  <15>\Top of debugger\                        ;
                                                            72  0032C                                                         ;
61  63  69  73  79  68  70  20  66  6F  20  70  6F  54  16  0032D P.ABS:   .ASCII  <22>\Top of physical memory\                 ;
                            79  72  6F  6D  65  6D  20  6C  0033C                                                         ;

                                                                           .EXTRN  DS$PRINTF

                                                                           .PSECT  CODE,NOWRT,  SHR,2

                                                  0FFC 00000 SHOWMEMORY:
                                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11     ; 0205
                             5B 00000000G   EF   9E 00002                  MOVAB   DS$GL_LOOKASIDE, R11                     ;
                             5A 00000000G   EF   9E 00009                  MOVAB   DS$GB_TYPECODE, R10                      ;
                             59 00000000G   EF   9E 00010                  MOVAB   DS$GL_BUFCNT, R9                         ;
                             58 00000000G   EF   9E 00017                  MOVAB   DS$GL_MEMSIZE, R8                        ;
                             57 00000000G   EF   9E 0001E                  MOVAB   DS$GA_LASTADR, R7                        ;
                             56 00000000G   9F   9E 00025                  MOVAB   @#DS$PRINTF, R6                          ;
                             55 00000000'   EF   9E 0002C                  MOVAB   FORMAT, R5                               ;
    50 00000000'   EF        04             00   EF 00033                  EXTZV   #0, #4, SHOWMEMORYFLAGS, R0              ; 0216
                             53             50   90 0003C                  MOVB    R0, MEMORYFLAGS                          ;
                                            03   12 0003F                  BNEQ    1$                                       ; 0218
                             53             01   88 00041                  BISB2   #1, MEMORYFLAGS                          ; 0220
                   03        53             03   E1 00044 1$:              BBC     #3, MEMORYFLAGS, 2$                      ; 0222
                             53             07   88 00048                  BISB2   #7, MEMORYFLAGS                          ; 0227
    50 0000FE03    9F        01             04   EF 0004B 2$:              EXTZV   #4, #1, @#^X0000FE03, R0                 ; 0230
                             54             50   90 00054                  MOVB    R0, ONLINE                               ;
                             6A             16   90 00057                  MOVB    #22, DS$GB_TYPECODE                      ; 0231
                             35             53   E9 0005A                  BLBC    MEMORYFLAGS, 3$                          ; 0233
                                    009F    C5   9F 0005D                  PUSHAB  T_RESERVED                               ; 0239
```

```
                                    7E  D4 00061          CLRL    -(SP)                                          ;
                                    55  DD 00063          PUSHL   R5                                             ;
                              66    03  FB 00065          CALLS   #3, DS$PRINTF                                  ;
                              27    53  E9 00068          BLBC    MEMORYFLAGS, 3$                                ; 0241
                                01C7  C5  9F 0006B          PUSHAB  P.AAY                                          ; 0247
                            00C00000G EF  9F 0006F          PUSHAB  DS$A_PRGBGN                                    ;
                                    55  DD 00075          PUSHL   R5                                             ;
                              66    03  FB 00077          CALLS   #3, DS$PRINTF                                  ;
                              15    53  E9 0007A          BLBC    MEMORYFLAGS, 3$                                ; 0249
                                FF5B  30  0007D          BSBW    DSR$CHECKLOAD                                  ;
                              0F    50  E9 00080          BLBC    R0, 3$                                         ;
                                01E6  C5  9F 00083          PUSHAB  P.AAZ                                          ; 0255
                            000002^4  9F  DD 00087          PUSHL   @#^X00000214                                   ;
                                    55  DD 0008D          PUSHL   R5                                             ;
                              66    03  FB 0008F          CALLS   #3, DS$PRINTF                                  ;
                    2E        53    01  E1 00092 3$:      BBC     #1, MEMORYFLAGS, 4$                            ; 0257
                                FF42  30  00096          BSBW    DSR$CHECKLOAD                                  ; 0258
                              28    50  E9 00099          BLBC    R0, 4$                                         ;
                                01FF  C5  9F 0009C          PUSHAB  P.ABA                                          ; 0266
                                01AE  C5  9F 000A0          PUSHAB  T_BUFFERSPACE                                  ;
        50 00000214   9F 0000FA00   8F  C3 000A4          SUBL3   #64000, @#^X00000214, R0                       ;
        51            50 00000400   8F  C7 000B0          DIVL3   #1024, R0, R1                                  ;
                      7E            51  CE 000B8          MNEGL   R1, -(SP)                                      ;
                      7E            50  CE 000BB          MNEGL   R0, -(SP)                                      ;
                                 75   A5  9F 000BE          PUSHAB  FORMAT5                                        ;
                              66    05  FB 000C1          CALLS   #5, DS$PRINTF                                  ;
                              04    53  E8 000C4 4$:      BLBS    MEMORYFLAGS, 5$                                ; 0268
                    0E        53    02  E1 000C7          BBC     #2, MEMORYFLAGS, 6$                            ;
                                0210  C5  9F 000CB 5$:      PUSHAB  P.ABB                                          ; 0274
                              7E  FA00  8F  3C 000CF          MOVZWL  #64000, -(SP)                                  ;
                                    55  DD 000D4          PUSHL   R5                                             ;
                              66    03  FB 000D6          CALLS   #3, DS$PRINTF                                  ;
                              04    53  E8 000D9 6$:      BLBS    MEMORYFLAGS, 7$                                ; 0276
                    0E        53    02  E1 000DC          BBC     #2, MEMORYFLAGS, 8$                            ; 0277
                                0221  C5  9F 000E0 7$:      PUSHAB  P.ABC                                          ; 0283
                              7E  FE00  8F  3C 000E4          MOVZWL  #65024, -(SP)                                  ;
                                    55  DD 000E9          PUSHL   R5                                             ;
                              66    03  FB 000EB          CALLS   #3, DS$PRINTF                                  ;
                              26    53  E9 000EE 8$:      BLBC    MEMORYFLAGS, 9$                                ; 0285
                                00AD  C5  9F 000F1          PUSHAB  T_DS                                           ; 0292
                                022D  C5  9F 000F5          PUSHAB  P.ABD                                          ;
                            00010000  8F  DD 000F9          PUSHL   #65536                                         ;
                                  1C  A5  9F 000FF          PUSHAB  FORMAT2                                        ;
                              66    04  FB 00102          CALLS   #4, DS$PRINTF                                  ;
                              0F    53  E9 00105          BLBC    MEMORYFLAGS, 9$                                ; 0294
                                023B  C5  9F 00108          PUSHAB  P.ABE                                          ; 0300
                            00000000G EF  9F 0010C          PUSHAB  AX_BUFF                                        ;
                                    55  DD 00112          PUSHL   R5                                             ;
                              66    03  FB 00114          CALLS   #3, DS$PRINTF                                  ;
                    03        53    02  E0 00117 9$:      BBS     #2, MEMORYFLAGS, 11$                           ; 0302
                                01A7  31  0011B 10$:      BRW     12$                                            ;
                              FA    54  E8 0011E 11$:     BLBS    ONLINE, 10$                                    ;
                                0258  C5  9F 00121          PUSHAB  P.ABF                                          ; 0309
                            00000000G EF  9F 00125          PUSHAB  PHD_BASE                                       ;
                                    55  DD 0012B          PUSHL   R5                                             ;
                              66    03  FB 0012D          CALLS   #3, DS$PRINTF                                  ;
                                025C  C5  9F 00130          PUSHAB  P.ABG                                          ; 0314
```

```
                00000000G  EF  9F 00134          PUSHAB  PCB_BASE                        ;
                           55  DD 0013A          PUSHL   R5                              ;
                66         03  FB 0013C          CALLS   #3, DS$PRINTF                   ;
                     0269  C5  9F 0013F          PUSHAB  P.ABH                           ; 0319
                00000000G  EF  9F 00143          PUSHAB  DS$AX_JIB                       ;
                           55  DD 00149          PUSHL   R5                              ;
                66         03  FB 0014B          CALLS   #3, DS$PRINTF                   ;
                     026D  C5  9F 0014E          PUSHAB  P.ABI                           ; 0324
                00000000G  EF  9F 00152          PUSHAB  DS$AX_ARB                       ;
                           55  DD 00158          PUSHL   R5                              ;
                66         03  FB 0015A          CALLS   #3, DS$PRINTF                   ;
                     0271  C5  9F 0015D          PUSHAB  P.ABJ                           ; 0329
                00000000G  EF  9F 00161          PUSHAB  DS$AX_SOFTPCB                   ;
                           55  DD 00167          PUSHL   R5                              ;
                66         03  FB 00169          CALLS   #3, DS$PRINTF                   ;
                     027E  C5  9F 0016C          PUSHAB  P.ABK                           ; 0334
                00000000G  EF  9F 00170          PUSHAB  SCB_BASE                        ;
                           55  DD 00176          PUSHL   R5                              ;
                66         03  FB 00178          CALLS   #3, DS$PRINTF                   ;
                     009F  C5  9F 0017B          PUSHAB  T_RESERVED                      ; 0339
                00000000G  EF  9F 0017F          PUSHAB  STACK_BASE                      ;
                           55  DD 00185          PUSHL   R5                              ;
                66         03  FB 00187          CALLS   #3, DS$PRINTF                   ;
                52         00  DB 0018A          MFPR    #0, TEMP                        ; 0340
                     00FA  C5  9F 0018D          PUSHAB  T_SP                            ; 0346
                     00C3  C5  9F 00191          PUSHAB  T_KERNEL                        ;
                           52  DD 00195          PUSHL   TEMP                            ;
                       1C  A5  9F 00197          PUSHAB  FORMAT2                         ;
                66         04  FB 0019A          CALLS   #4, DS$PRINTF                   ;
                     00EE  C5  9F 0019D          PUSHAB  T_STACK                         ; 0352
                     00C3  C5  9F 001A1          PUSHAB  T_KERNEL                        ;
                00000000G  EF  9F 001A5          PUSHAB  KSTKPTR                         ;
                       3B  A5  9F 001AB          PUSHAB  FORMAT3                         ;
                66         04  FB 001AE          CALLS   #4, DS$PRINTF                   ;
                52         04  DB 001B1          MFPR    #4, TEMP                        ; 0353
                     00FA  C5  9F 001B4          PUSHAB  T_SP                            ; 0359
                     00CA  C5  9F 001B8          PUSHAB  T_INTERRUPT                     ;
                           52  DD 001BC          PUSHL   TEMP                            ;
                       1C  A5  9F 001BE          PUSHAB  FORMAT2                         ;
                66         04  FB 001C1          CALLS   #4, DS$PRINTF                   ;
                     00EE  C5  9F 001C4          PUSHAB  T_STACK                         ; 0365
                     00CA  C5  9F 001C8          PUSHAB  T_INTERRUPT                     ;
                00000000G  EF  9F 001CC          PUSHAB  ISTKPTR                         ;
                       3B  A5  9F 001D2          PUSHAB  FORMAT3                         ;
                66         04  FB 001D5          CALLS   #4, DS$PRINTF                   ;
                52         01  DB 001D8          MFPR    #1, TEMP                        ; 0366
                     00FA  C5  9F 001DB          PUSHAB  T_SP                            ; 0372
                     00D4  C5  9F 001DF          PUSHAB  T_EXECUTIVE                     ;
                           52  DD 001E3          PUSHL   TEMP                            ;
                       1C  A5  9F 001E7          PUSHAB  FORMAT2                         ;
                66         04  FB 001E8          CALLS   #4, DS$PRINTF                   ;
                     00EE  C5  9F 001EB          PUSHAB  T_STACK                         ; 0378
                     00D4  C5  9F 001EF          PUSHAB  T_EXECUTIVE                     ;
                00000000G  EF  9F 001F3          PUSHAB  ESTKPTR                         ;
                       3B  A5  9F 001F9          PUSHAB  FORMAT3                         ;
                66         04  FB 001FC          CALLS   #4, DS$PRINTF                   ;
                52         02  DB 001FF          MFPR    #2, TEMP                        ; 0379
```

```
                                00FA   C5  9F 00202          PUSHAB  T_SP                                    ; 0385
                                00DE   C5  9F 00206          PUSHAB  T_SUPERVISOR                            ;
                                       52  DD 0020A          PUSHL   TEMP                                    ;
                                  1C   A5  9F 0020C          PUSHAB  FORMAT2                                 ;
                              66       04  FB 0020F          CALLS   #4, DS$PRINTF                           ;
                                00EE   C5  9F 00212          PUSHAB  T_STACK                                 ; 0391
                                00DE   C5  9F 00216          PUSHAB  T_SUPERVISOR                            ;
                           00000000G   EF  9F 0021A          PUSHAB  SSTKPTR                                 ;
                                  3B   A5  9F 00220          PUSHAB  FORMAT3                                 ;
                              66       04  FB 00223          CALLS   #4, DS$PRINTF                           ;
                              52       03  DB 00226          MFPR    #3, TEMP                                ; 0392
                                00FA   C5  9F 00229          PUSHAB  T_SP                                    ; 0398
                                00E9   C5  9F 0022D          PUSHAB  T_USER                                  ;
                                       52  DD 00231          PUSHL   TEMP                                    ;
                                  1C   A5  9F 00233          PUSHAB  FORMAT2                                 ;
                              66       04  FB 00236          CALLS   #4, DS$PRINTF                           ;
                                00EE   C5  9F 00239          PUSHAB  T_STACK                                 ; 0404
                                00E9   C5  9F 0023D          PUSHAB  T_USER                                  ;
                           00000000G   EF  9F 00241          PUSHAB  USTKPTR                                 ;
                                  3B   A5  9F 00247          PUSHAB  FORMAT3                                 ;
                              66       04  FB 0024A          CALLS   #4, DS$PRINTF                           ;
                              52       08  DB 0024D          MFPR    #8, TEMP                                ; 0405
                                010B   C5  9F 00250          PUSHAB  T_BASE                                  ; 0411
                                00FE   C5  9F 00254          PUSHAB  T_P0                                    ;
          7E        52          1E       00  EF 00258          EXTZV   #0, #30, TEMP, -(SP)                    ;
                                  1C   A5  9F 0025D          PUSHAB  FORMAT2                                 ;
                              66       04  FB 00260          CALLS   #4, DS$PRINTF                           ;
                              52       09  DB 00263          MFPR    #9, TEMP                                ; 0412
                                011C   C5  9F 00266          PUSHAB  T_LENGTH                                ; 0418
                                00FE   C5  9F 0026A          PUSHAB  T_P0                                    ;
                                       52  DD 0026E          PUSHL   TEMP                                    ;
                                  3B   A5  9F 00270          PUSHAB  FORMAT3                                 ;
                              66       04  FB 00273          CALLS   #4, DS$PRINTF                           ;
                              52       0A  DB 00276          MFPR    #10, TEMP                               ; 0419
                                010B   C5  9F 00279          PUSHAB  T_BASE                                  ; 0425
                                0101   C5  9F 0027D          PUSHAB  T_P1                                    ;
          7E        52          1E       00  EF 00281          EXTZV   #0, #30, TEMP, -(SP)                    ;
                                  1C   A5  9F 00286          PUSHAB  FORMAT2                                 ;
                              66       04  FB 00289          CALLS   #4, DS$PRINTF                           ;
                              52       0B  DB 0028C          MFPR    #11, TEMP                               ; 0426
                                011C   C5  9F 0028F          PUSHAB  T_LENGTH                                ; 0432
                                0101   C5  9F 00293          PUSHAB  T_P1                                    ;
                                       52  DD 00297          PUSHL   TEMP                                    ;
                                  3B   A5  9F 00299          PUSHAB  FORMAT3                                 ;
                              66       04  FB 0029C          CALLS   #4, DS$PRINTF                           ;
                              52       0C  DB 0029F          MFPR    #12, TEMP                               ; 0433
                                010B   C5  9F 002A2          PUSHAB  T_BASE                                  ; 0439
                                0104   C5  9F 002A6          PUSHAB  T_SYSTEM                                ;
                                       52  DD 002AA          PUSHL   TEMP                                    ;
                                  1C   A5  9F 002AC          PUSHAB  FORMAT2                                 ;
                              66       04  FB 002AF          CALLS   #4, DS$PRINTF                           ;
                              52       0D  DB 002B2          MFPR    #13, TEMP                               ; 0440
                                011C   C5  9F 002B5          PUSHAB  T_LENGTH                                ; 0446
                                0104   C5  9F 002B9          PUSHAB  T_SYSTEM                                ;
                                       52  DD 002BD          PUSHL   TEMP                                    ;
                                  3B   A5  9F 002BF          PUSHAB  FORMAT3                                 ;
                              66       04  FB 002C2          CALLS   #4, DS$PRINTF                           ;
```

```
                                04              53  E8 002C5 12$:    BLBS    MEMORYFLAGS, 13$                          ; 0449
                            23  53              02  E1 002C8         BBC     #2, MEMORYFLAGS, 14$                      ;
                                        0282    C5  9F 002CC 13$:    PUSHAB  P.ABL                                     ; 0456
                                                6B  DD 002D0         PUSHL   DS$GL_LOOKASIDE                           ;
                                                55  DD 002D2         PUSHL   R5                                        ;
                                66              03  FB 002D4         CALLS   #3, DS$PRINTF                             ;
                             00000000G  EF              6B  D1 002D7         CMPL    DS$GL_LOOKASIDE, EXE$GL_SPLITADR  ; 0458
                                                0F  13 002DE         BEQL    14$                                       ;
                                        02A2    C5  9F 002E0         PUSHAB  P.ABM                                     ; 0464
                                    00000000G   EF  DD 002E4         PUSHL   EXE$GL_SPLITADR                           ;
                                                55  DD 002EA         PUSHL   R5                                        ;
                                66              03  FB 002EC         CALLS   #3, DS$PRINTF                             ;
                                04              53  E8 002EF 14$:    BLBS    MEMORYFLAGS, 15$                          ; 0468
                            20  53              02  E1 002F2         BBC     #2, MEMORYFLAGS, 16$                      ; 0469
                                        00AD    C5  9F 002F6 15$:    PUSHAB  T_DS                                      ; 0477
                                        02BF    C5  9F 002FA         PUSHAB  P.ABN                                     ;
                            50  67 000003FF      8F  C1 002FE         ADDL3   #1023, DS$GA_LASTADR, R0                  ;
                            7E  50 00000400     8F  C7 00306         DIVL3   #1024, R0, -(SP)                          ;
                                                67  DD 0030E         PUSHL   DS$GA_LASTADR                             ;
                                            75  A5  9F 00310         PUSHAB  FORMAT5                                   ;
                                66              05  FB 00313         CALLS   #5, DS$PRINTF                             ;
                            48  53              01  E1 00316 16$:    BBC     #1, MEMORYFLAGS, 18$                      ; 0479
                                67              68  D1 0031A         CMPL    DS$GL_MEMSIZE, DS$GA_LASTADR              ; 0480
                                                1C  15 0031D         BLEQ    17$                                       ;
                                        02D0    C5  9F 0031F         PUSHAB  P.ABO                                     ; 0488
                                        01AE    C5  9F 00323         PUSHAB  T_BUFFERSPACE                             ;
                            50  68              67  C3 00327         SUBL3   DS$GA_LASTADR, DS$GL_MEMSIZE, R0          ;
                            7E  50 00000400     8F  C7 0032B         DIVL3   #1024, R0, -(SP)                          ;
                                                50  DD 00333         PUSHL   R0                                        ;
                                            75  A5  9F 00335         PUSHAB  FORMAT5                                   ;
                                66              05  FB 00338         CALLS   #5, DS$PRINTF                             ;
                            23  53              01  E1 0033B 17$:    BBC     #1, MEMORYFLAGS, 18$                      ; 0490
                                        013B    C5  9F 0033F         PUSHAB  T_P0_BUFCNT                               ; 0496
                                                69  DD 00343         PUSHL   DS$GL_BUFCNT                              ;
                                                55  DD 00345         PUSHL   R5                                        ;
                                66              03  FB 00347         CALLS   #3, DS$PRINTF                             ;
                                        0160    C5  9F 0034A         PUSHAB  T_P1_BUFCNT                               ; 0501
                                          04    A9  DD 0034E         PUSHL   DS$GL_BUFCNT+4                            ;
                                                55  DD 00351         PUSHL   R5                                        ;
                                66              03  FB 00353         CALLS   #3, DS$PRINTF                             ;
                                        0185    C5  9F 00356         PUSHAB  T_S0_BUFCNT                               ; 0506
                                          08    A9  DD 0035A         PUSHL   DS$GL_BUFCNT+8                            ;
                                                55  DD 0035D         PUSHL   R5                                        ;
                                66              03  FB 0035F         CALLS   #3, DS$PRINTF                             ;
                        2C 0000FE03     9F      02  E1 00362 18$:    BBC     #2, @#^X0000FE03, 20$                     ; 0509
                                        02E1    C5  9F 0036A         PUSHAB  P.ABP                                     ; 0514
                                    00000000G   EF  DD 0036E         PUSHL   DEBUG_LO                                  ;
                                                55  DD 00374         PUSHL   R5                                        ;
                                66              03  FB 00376         CALLS   #3, DS$PRINTF                             ;
                                08              54  E8 00379         BLBS    ONLINE, 19$                               ; 0515
                                        02F4    C5  9F 0037C         PUSHAB  P.ABQ                                     ; 0518
                                        0092    C5  9F 00380         PUSHAB  FORMAT6                                   ;
                                66              02  FB 00384         CALLS   #2, DS$PRINTF                             ;
                                        031D    C5  9F 00387 19$:    PUSHAB  P.ABR                                     ; 0521
                                    00000000G   EF  DD 0038B         PUSHL   DEBUG_HI                                  ;
                                                55  DD 00391         PUSHL   R5                                        ;
                                66              03  FB 00393         CALLS   #3, DS$PRINTF                             ;
```

```
                                14              54  E8 00396 20$:    BLBS    ONLINE, 21$                                     ; 0524
                                          032D  C5  9F 00399         PUSHAB  P.ABS                                           ; 0531
                          7E    68 00000400   8F  C7 0039D         DIVL3   #1024, DS$GL_MEMSIZE, -(SP)                     ;
                                              68  DD 003A5         PUSHL   DS$GL_MEMSIZE                                   ;
                                          5B  A5  9F 003A7         PUSHAB  FORMAT4                                         ;
                                66            04  FB 003AA         CALLS   #4, DS$PRINTF                                   ;
                                              6A  94 003AD 21$:    CLRB    DS$GB_TYPECODE                                  ; 0532
                                                  04 003AF         RET                                                     ; 0533

; Routine Size:  944 bytes,    Routine Base:  CODE + 0025

;   0534  1
```

```
;    0535  1      !                                                                                    BEGIN   [05]
;    0536  1      %SBTTL 'VRSETMEM'
;    0537  1
;    0538  1      GLOBAL ROUTINE VRSETMEM (CLI_BASE) : JSB_CLI =
;    0539  1
;    0540  1      !++
;    0541  1      ! FUNCTIONAL DESCRIPTION:
;    0542  1      !
;    0543  1      !       This routine is called from the CLI module. It sets the value of
;    0544  1      !       DS$GL_SET_MEMSIZE to the value specified, remaps memory, and goes
;    0545  1      !       to command level (DS>).
;    0546  1      !
;    0547  1      ! CALLER(S):
;    0548  1      !
;    0549  1      !       CLI
;    0550  1      !
;    0551  1      ! FORMAL PARAMETERS:
;    0552  1      !
;    0553  1      !       R2 - Points to the CLI data table base.
;    0554  1      !
;    0555  1      ! IMPLICIT INPUTS:
;    0556  1      !
;    0557  1      !       CLI_BASE [CLI$L_DATA]  - Indicates the new value of memory.
;    0558  1      !
;    0559  1      ! IMPLICIT OUTPUTS:
;    0560  1      !
;    0561  1      !       New value for DS$GL_SET_MEMSIZE.
;    0562  1      !
;    0563  1      ! COMPLETION CODES:
;    0564  1      !
;    0565  1      !       None
;    0566  1      !
;    0567  1      ! SIDE EFFECTS:
;    0568  1      !
;    0569  1      !       Executes user's cleanup code.
;    0570  1      !--
;    0571  1
;    0572  2      BEGIN
;    0573  2
;    0574  2          EXTERNAL ROUTINE
;    0575  2              BEGIN_BLISS : INT,
;    0576  2              DSR$MMENABLE : JSB_0,
;    0577  2              DS_CLEANUP : JSB_NONE,
;    0578  2              INI$LOAD_DEVICE : JSB_NONE,                                  !                  [06]
;    0579  2              INI$PTABLE : JSB_NONE,
;    0580  2              MAPMEM : JSB_NONE,
;    0581  2              SCRIPT$FLUSH : JSB_NONE;
;    0582  2
;    0583  2          LOCAL
;    0584  2              WASSET;
;    0585  2
;    0586  2          MAP
;    0587  2              CLI_BASE : REF BLOCK [, BYTE];
;    0588  2
;    0589  3          IF (.CLI_BASE [CLI$L_DATA] LSS 0)
;    0590  3              THEN $DS_PRINTF (%ASCIC (%STRING ('?? Error, negative number of pages specified!!', %CHAR (CR, LF))))
;    0591  3              ELSE BEGIN
```

```
;    0592  3                    DS$GL_SET_MEMSIZE = .CLI_BASE [CLI$L_DATA];          ! Set the new value of memory
;    0593  3                    SCRIPT$FLUSH ();                                     ! Flush scripts if console command
;    0594  3                    DS_CLEANUP ();                                       ! Execute user's cleanup code
;    0595  3                    WASSET = DSR$MMENABLE (0);                           ! Turn off memory management
;    0596  3                    MAPMEM ();                                           ! Remap memory with new value
;    0597  3                    IF .WASSET EQL SS$_WASSET THEN DSR$MMENABLE (1);     ! Reset memory management if on before
;    0598  3                    INI$PTABLE ();                                       ! Initialize P-tables
;    0599  3                    INI$LOAD_DEVICE ();                                  ! ATT console and boot devices          [06]
;    0600  3                    BEGIN_BLISS ()                                       ! Reset and go to DS>
;    0601  3                END
;    0602  1     END;                                                                ! End                                   [05]


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61 67 65 6E 20 2C 72 6F 72 72 45 20 3F 3F 30  00344 P.ABT:  .ASCII  \0?? Error, negative number of pages spec\  ;
20 66 6F 20 72 65 62 6D 75 6E 20 65 76 69 74  00353                                                          ;
           63 65 70 73 20 73 65 67 61 70 69   00362                                                          ;
              0A 0D 21 21 64 65 69 66 69 66   0036C  .ASCII  \ified!!\<13><10>                               ;

                                                        .EXTRN  BEGIN_BLISS, DSR$MMENABLE
                                                        .EXTRN  DS_CLEANUP, INI$LOAD_DEVICE
                                                        .EXTRN  INI$PTABLE, MAPMEM
                                                        .EXTRN  SCRIPT$FLUSH

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                    52  DD 00000 VRSETMEM::
                                                    PUSHL   R2                                          ; 0538
                                 1C     A2  D5 00002        TSTL    28(CLI_BASE)                        ; 0589
                                        0F  18 00005        BGEQ    1$                                  ;
                           00000000'    EF  9F 00007        PUSHAB  P.ABT                               ; 0590
                00000000G   9F          01  FB 0000D        CALLS   #1, @#DS$PRINT                      ;
                                        45  11 00014        BRB     3$                                  ;
                00000000G   EF     1C   A2  D0 00016 1$:    MOVL    28(CLI_BASE), DS$GL_SET_MEMSIZE     ; 0592
                           00000000G    EF  16 0001E        JSB     SCRIPT$FLUSH                        ; 0593
                           00000000G    EF  16 00024        JSB     DS_CLEANUP                          ; 0594
                                        50  D4 0002A        CLRL    R0                                  ; 0595
                           00000000G    EF  16 0002C        JSB     DSR$MMENABLE                        ;
                                    52  50  D0 00032        MOVL    R0, WASSET                          ;
                           00000000G    EF  16 00035        JSB     MAPMEM                              ; 0596
                                    09  52  D1 0003B        CMPL    WASSET, #9                          ; 0597
                                        09  12 0003E        BNEQ    2$                                  ;
                                    50  01  D0 00040        MOVL    #1, R0                              ;
                           00000000G    EF  16 00043        JSB     DSR$MMENABLE                        ;
                           00000000G    EF  16 00049 2$:    JSB     INI$PTABLE                          ; 0598
                           00000000G    EF  16 0004F        JSB     INI$LOAD_DEVICE                     ; 0599
                           00000000G    EF  16 00055        JSB     BEGIN_BLISS                         ; 0600
                                        04  BA 0005B 3$:    POPR    #^M<R2>                             ; 0602
                                            05 0005D        RSB                                         ;

; Routine Size:  94 bytes,    Routine Base:  CODE + 03D5

;    0603  1
;    0604  1     End                                         ! End of module
```

```
;  0605  0     Eludom



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  DATA                          885  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                            1  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                         1075  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                               Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent     Mapped      Time
;
;  DRB1:[DS.WORK]DIAG.L32;265          784        11         1         85       00:00.2
;  DRB1:[DS.WORK]DS.L32;159            653         6         0         42       00:00.2
;  SYS$SYSROOT:[SYSLIB]LIB.L32;7     18017        15         0        975       00:04.6



;                                       COMMAND QUALIFIERS

;       BLISS/NOOBJECT/LIST=[DS.LIS]/OPTIMIZE=(LEVEL:3,SPACE)/DEBUG/TRACE SHOWMEMORY

; Size:          1075 code + 886 data bytes
; Run Time:         00:24.9
; Elapsed Time:     00:49.0
; Lines/CPU Min:     1460
; Lexemes/CPU-Min: 18569
; Memory Used:  276 pages
; Compilation Complete
```

Table of contents

(3) 178 Declarations
(3) 269 VRStart and VRReStart Routines
(3) 700 VRAbort and DSX$Abort Routines
(4) 842 VRSupport Routines
(4) 901 VRShowSections Routine
(4) 963 DSI$TIME_AST Routine

```
                                        .TITLE  START   *** START Handle START command
                                        .IDENT  /07-41/
                                        .NoShow Conditionals

                                ;
                                ; Copyright (c) 1977, 1981, 1982, 1983, 1984
                                ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                                ;
                                ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                                ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                                ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                                ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                                ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                                ; REMAIN IN DEC.
                                ;
                                ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                ; CORPORATION.
                                ;
                                ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                                ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                ;
                                ;++
                                ; FACILITY:      VAX DIAGNOSTIC SUPERVISOR.
                                ;
                                ; ABSTRACT:      CONTAINS ALL PROGRAM STARTING AND STOPPING FUNCTIONS:
                                ;                         "START" COMMAND SUBROUTINE.
                                ;                         "CONTINUE" COMMAND SUBROUTINE.
                                ;                         "ABORT" COMMAND SUBROUTINE.
                                ;                         $DS_ABORT DIAGNOSTIC SERVICE PROCEDURE.
                                ;
                                ; ENVIRONMENT:
                                ;
                                ; AUTHOR:        KEN CHAPMAN     10-NOV-77       VERSION 01.
```

```
                        0000     37 ; MODIFIED BY:
                        0000     38 ;
                        0000     39 ;       01      DSPR #25 - CONTINUE FROM BREAKPOINT PROBLEMS.
                        0000     40 ;       02      DSPR #42 - APT FUNCTION UPDATE.
                        0000     41 ;       03      FIX TO AVOID "BUG CHECK" WHEN PROCEEDING FROM AN EXCEPTION.
                        0000     42 ;
                        0000     43 ;               KEN CHAPMAN     08-DEC-77       VERSION 03 (ESSAA-3.05).
                        0000     44 ;       04      ADD NEW-LINE TO END OF PROGRAM TITLE MESSAGE.
                        0000     45 ;       05      DSPR #4, #5 - ADDED CONTINUE AND ABORT MESSAGES.
                        0000     46 ;
                        0000     47 ;               N. HOWGATE      15-FEB-78       VERSION 04 (ESSAA-4.00).
                        0000     48 ;       06      QIO SUPPORT
                        0000     49 ;
                        0000     50 ;               KEN CHAPMAN     20-MAR-78       VERSION 05 (ESSAA-4.01).
                        0000     51 ;       07      ALLOW CONTINUE FROM ANY BREAKPOINT OR ^C.
                        0000     52 ;       08      ADD TEST COUNT TO PROGRAM TITLE MESSAGE.
                        0000     53 ;       09      CLEAR TIMER VECTOR IN STARTUP SEQUENCE.
                        0000     54 ;               Roger Riggs     12-Jun-78       VERSION 06 (ESSAA-4.03).
                        0000     55 ;       10      Standardizing program related messages.
                        0000     56 ;               NICK HOWGATE    23-JUN-78
                        0000     57 ;       11      CORRECT 'APT' ABORT PROBLEM
                        0000     58 ;               Roger Riggs             07-Aug-78
                        0000     59 ;       12      Install test number range checking, also update DS$GL_NUMTEST.
                        0000     60 ;               fix so DS$M_DONFLG is set by DS_CLEANUP and not checked
                        0000     61 ;               before calling it.
                        0000     62 ;       13      Changes reflected by new structure of basic loop.
                        0000     63 ;               i.e. START and RESTART commands call test program directly
                        0000     64 ;               Roger Riggs             23-OCT-1978     VERSION 07 (ESSAA-5.01)
                        0000     65 ;       14      Changes to VRSTART to support new QIO initialization.
                        0000     66 ;       15      Added code to verify that this program is compatible with
                        0000     67 ;               this supervisor, via ENV$V_SUPER, Value must match
                        0000     68 ;               ENV$_SUPER, in DIAG. Updated when supervisor interface
                        0000     69 ;               changes REQUIRE diagnostic changes.
                        0000     70 ;               Roger Riggs     4-Sept-1979
                        0000     71 ;       17      Removed check of APT flag during start command
                        0000     72 ;               Roger Riggs, 14-Feb-1980, Vervsion 5.2
                        0000     73 ;       18      Changed so program will not be started if QIO initialization
                        0000     74 ;               fails, QIO has already printed the error message.
                        0000     75 ;               Roger Riggs, 12-Mar-1980, Version 5.3
                        0000     76 ;       19      In VRABORT or DSX$ABORT change mode back to kernel before
                        0000     77 ;               doing cleanup but after the abort message is printed
                        0000     78 ;               Roger Riggs, 5-Jun-1980, Version 5.4
                        0000     79 ;
                        0000     80 ;       20      Added code to change mode and stack back to kernel
                        0000     81 ;               before calling cleanup as a result of start.
```

```
                     0000     83 ;        21      - dave butenhof, 09-Sep-1981, version 6.-
                     0000     84 ;                It is desirable, under APT, and if program error had
                     0000     85 ;                occured, to have ABORT leave machine state intact, for
                     0000     86 ;                purposes of isolating error.  Add special code to skip
                     0000     87 ;                INIT_CONTEXT and CLEANUP calls if in APT mode and error
                     0000     88 ;                flag set.
                     0000     89 ;
                     0000     90 ;        22      - Marion Baggett 80-oct-1981 Version 6.-
                     0000     91 ;                Added new entry point to support command SHOW SUPPORT
                     0000     92 ;
                     0000     93 ;        23      - Jack Stansbury, 21-Oct-1981, Version 6.-
                     0000     94 ;                Added calls to QA$MAIN for the QA enhancement.
                     0000     95 ;                Also added .LIBRARY statements for $DS and $DIAG.
                     0000     96 ;
                     0000     97 ;        24      - Jack Stansbury, 22-Oct-1981, Version 6.-
                     0000     98 ;                Fixed truncation errors.
                     0000     99 ;
                     0000    100 ;        25      - Jack Stansbury, 26-Oct-1981, Version 6.-
                     0000    101 ;                Changed most of the error messages to mixed case
                     0000    102 ;                (as per government affirmative action regulations)!
                     0000    103 ;
                     0000    104 ;        26      - Dave Butenhof, 19-Nov-1981, version 6.-
                     0000    105 ;                Add type codes to printouts
                     0000    106 ;
                     0000    107 ;        27      - Jack Stansbury, 16-Nov-1981, Version 6.5
                     0000    108 ;                Added the VRSHOWSECTIONS routine.
                     0000    109 ;
                     0000    110 ;        28      - Jack Stansbury, 21-Nov-1981, Version 6.5
                     0000    111 ;                Commented out three of the calls to QA_MAIN because it
                     0000    112 ;                appears they will not be needed for the 6.5 DS version.
                     0000    113 ;
                     0000    114 ;        29      - Jack Stansbury, 5-Jan-1982, Version 6.6
                     0000    115 ;                Added comments to the START routine in preparation for the
                     0000    116 ;                version 30 changes. Also took out the commented-out calls
                     0000    117 ;                to QA_MAIN since I will not be needing them.
                     0000    118 ;
                     0000    119 ;        30      - Jack Stansbury, 6-Jan-1982, Version 6.6
                     0000    120 ;                Added code and moved other code around in the START routine so
                     0000    121 ;                as to facilitate the "QA Start Loop".
                     0000    122 ;
                     0000    123 ;        31      - Jack Stansbury, 15-Jan-1982, Version 6.6
                     0000    124 ;                Changed the FMT_PROG header line so that it would print the
                     0000    125 ;                number of tests on the first line. Only the time appears on the
                     0000    126 ;                second line. This is in keeping with the EndPass message and
                     0000    127 ;                the End-Of-Looping message (in EndSub).
                     0000    128 ;
                     0000    129 ;        32      - Jack Stansbury, 20-Jan-1982, Version 6.6
                     0000    130 ;                Added code to not print the header line if the QA Loop on
                     0000    131 ;                Subtest check was executing and if a flag bit had been cleared.
                     0000    132 ;                This prevents the header line from being output once for every
                     0000    133 ;                subtest in the diagnostic when the Loop on Subtest check
                     0000    134 ;                routine is executing.
                     0000    135 ;
                     0000    136 ;        33      Marion Baggett, 9-Apr-1982, Version 6.7
                     0000    137 ;                Added .NoShow Conditionals after .IDENT , ordered .LIBRARY
                     0000    138 ;                statements. Change DS$PRINTF statement to use type codes
                     0000    139 ;
```

```
0000   140 ;        34      Richard Brown, 30-June-82, Version 6.8
0000   141 ;                Addition of call to MAP_DEBUGGER so repeated STARTs will
0000   142 ;                not wipe out the debugger once it has been loaded.
0000   143 ;                Also, addition of loading debugger before diagnostic is
0000   144 ;                started, if proper flag is set.
0000   145 ;
0000   146 ;        35      Marion Baggett, 24-Sept-82  Version 6.9
0000   147 ;                Fixed truncation error.
0000   148 ;
0000   149 ;        36      Bob Bergazzi    2-Nov-1982      Version 6.9
0000   150 ;                Changed VRSTART so that if we are loaded under APT do not
0000   151 ;                check the DSA$V_LOAD_DEBUGGER flag. APT sets this flag for
0000   152 ;                reasons long since lost. When running under APT, every
0000   153 ;                START would cause the supervisor to try and load the debugger
0000   154 ;                from the console device and cause APT to timeout.
0000   155 ;
0000   156 ;        37      Jack Stansbury, 8-Mar-1983, Version 6.11
0000   157 ;                Changed the abort routines to make sure they print the true
0000   158 ;                (if there is one) user PC.
0000   159 ;
0000   160 ;        38      John Ciukaj     3-May-1983      Version 6.11
0000   161 ;                - Alter Start routine so that an error does not occur
0000   162 ;                  when an ambiguous section name is found AND the section name
0000   163 ;                  is equivalent to the name entered.
0000   164 ;
0000   165 ;        39      Bob Bergazzi    May 16, 1983    Version 6.11
0000   166 ;                Changed the order of the .LIB statements.
0000   167 ;
0000   168 ;        40      Bob Bergazzi    Sep 21, 1983    Version 6.13
0000   169 ;                Added BIN_TIME quadword in PSECT WORK for START/TIME switch.
0000   170 ;                Just before call to DISPATCH, do a $SETIMR if /TIME was given.
0000   171 ;                Commented out for version 6.13 and 6.14, and 7.0
0000   172 ;
0000   173 ;        41      Bob Bergazzi    Jan. 26, 1984   Version 6.14
0000   174 ;                Fixed ABORT so it won't print out VMS system addresses as the
0000   175 ;                user PC.
0000   176 ;--
```

```
0000    178          .SBTTL  Declarations
0000    179 ;
0000    180 ; INCLUDE FILES:
0000    181 ;
0000    182          .LIBRARY        /Sys$library:lib/       ;                       [33]
0000    183          .LIBRARY        /$DS/                   ;                       [39]
0000    184          .LIBRARY        /$DIAG/                 ;                       [39]
0000    185
0000    186 ;
0000    187 ; MACROS:
0000    188 ;
0000    189
0000    190 ;
0000    191 ; EQUATED SYMBOLS:
0000    192 ;
0000    196          $DS_CFDEF
0000    197          $DS_CLIDEF
0000    198          $DS_DSADEF
0000    199          $DS_ENVDEF
0000    200          $DS_HDRDEF
0000    201          $DS_SCBDEF
0000    202          $PRDEF
0000    203          APTDEF
0000    204          CLIDEF
0000    205          DSFDEF
0000    206          DSQA                            ; DSQA$K constant definitions   [32]
0000    207          DS_QADEFS                       ; Other QA$K constant definitions [32]
0000    208          $ds_typedef                     ; Define type codes             [26]
0000    209
0000    213 ;        .PAGE
0000    214 ;        .PSECT  WORK, NOSHR, NOEXE, WRT, LONG                           ;       [40]
0000    215 ;
0000    216 ;BIN_TIME::      .BLKQ 1                 ; Holds converted 64-bit time   [40]
0000    217 ;                                        ; specified by user with /TIME  [40]
0000    218 ;BEGIN_TIME::    .BLKQ 1                 ; Holds absolute time when diag was [40]
0000    219 ;                                        ; started.                      [40]
0000    220 ;CTRLC_TIME::    .BLKQ 1                 ; Holds abs time at ^C.         [40]
```

```
                                        00000000    222          .PSECT  Data, Shr, NoExe, NoWrt, Byte
                                            0000    223
                                            0000    224          MODNAM  START
                        54 52 41 54 53 00'  0000        $MODULE:         .ASCIC  \START\
                                    05      0000
                                            0006    225
                                            0006    226 FMT_PROG:                                                   ;       [25]
61 72 67 6F 72 50 20 2E 2E 2F 21 00'        0006    227          .ASCIC  "!/.. Program: !AC, revision !UL.!UL, !UL test!%S,!/"-  ;       [31]
69 76 65 72 20 2C 43 41 21 20 3A 6D         0012
4C 55 21 2E 4C 55 21 20 6E 6F 69 73         001E
25 21 74 73 65 74 20 4C 55 21 20 2C         002A
25 21 20 74 61 20 20 20 2F 21 2C 53         0036
                        2F 21 2E 54         0042
                                    3F      0006
                                            C046    228                  "   at !%T.!/"                         ;       [31]
                                            0046    229
                                            0046    230 FMT_ABORT:                                                  ;       [25]
65 74 72 6F 62 41 20 2E 2E 2F 21 00'        0046    231          .ASCIC  \!/.. Aborted program at pass !UL, !AS, PC !XL.!/\      ;       [25]
74 61 20 6D 61 72 67 6F 72 70 20 64         0052
21 20 2C 4C 55 21 20 73 73 61 70 20         005E
21 2E 4C 58 21 20 43 50 20 2C 53 41         006A
                                    2F      0076
                                    30      0046
                                            0077    232
                                            0077    233 Fmt_Abort_No_PC:                                            ;       [37]
65 74 72 6F 62 41 20 2E 2E 2F 21 00'        0077    234          .ASCIC  \!/.. Aborted program at pass !UL, !AS.!/\      ;       [37]
74 61 20 6D 61 72 67 6F 72 70 20 64         0083
21 20 2C 4C 55 21 20 73 73 61 70 20         008F
                        2F 21 2E 53 41      009B
                                    28      0077
                                            00A0    235
                                            00A0    236 ;FMT_TIME_EXPIRED:
                                            00A0    237 ;        .ASCIC  "!/.. Time limit has expired at pass !UL, !AS, PC !XL.!/";      [40]
                                            00A0    238
                                            00A0    239 ;FMT_TIME_EXPIRED_NO_PC:
                                            00A0    240 ;        .ASCIC  "!/.. Time limit has expired at pass !UL, !AS.!/"       ;       [40]
                                            00A0    241
                                            00A0    242 FMT_TESTRANGE:                                              ;       [25]
21 20 79 6C 6E 4F 20 3F 3F 2F 21 00'        00A0    243          .ASCIC  \!/?? Only !UL test!%S, check test range.!/\           ;       [25]
20 2C 53 25 21 74 73 65 74 20 4C 55         00AC
72 20 74 73 65 74 20 6B 63 65 68 63         00B8
                  2F 21 2E 65 67 6E 61      00C4
                                    2A      00A0
```

```
                                        00CB   245 FMT_NOUNITS:                                                ;       [25]
73 74 69 6E 75 20 6F 4E 20 3F 3F 00'    00CB   246          .ASCIC  "?? No units to test, none selected with device type(s)!/";       [25]
6F 6E 20 2C 74 73 65 74 20 6F 74 20     00D7
20 64 65 74 63 65 6C 65 73 20 65 6E     00E3
20 65 63 69 76 65 64 20 68 74 69 77     00EF
           2F 21 29 73 28 65 70 79 74   00FB
                                 38     00CB
                                        0104   247
                                        0104   248 FMT_START:                                                 ;       [25]
6F 72 70 20 6F 4E 20 3F 3F 2F 21 00'    0104   249          .ASCIC  \!/?? No program loaded.!/\                      ;       [25]
2E 64 65 64 61 6F 6C 20 6D 61 72 67     0110
                              2F 21     011C
                                 19     0104
                                        011E   250
                                        C11E   251 FMT_ILLSEC:                                               ;       [25]
61 67 65 6C 6C 49 20 3F 3F 2F 21 00'    011E   252          .ASCIC  \!/?? Illegal section name, should be one of!/\          ;       [25]
61 6E 20 6E 6F 69 74 63 65 73 20 6C     012A
62 20 64 6C 75 6F 68 73 20 2C 65 6D     0136
      2F 21 66 6F 20 65 6E 6F 20 65     0142
                                 2D     011E
                                        014C   253
                                        014C   254 FMT_AMBSEC:                                               ;       [25]
75 67 69 62 6D 41 20 3F 3F 2F 21 00'    014C   255          .ASCIC  \!/?? Ambiguous section name, should be one of!/\        ;       [25]
20 6E 6F 69 74 63 65 73 20 73 75 6F     0158
64 6C 75 6F 68 73 20 2C 65 6D 61 6E     0164
2F 21 66 6F 20 65 6E 6F 20 65 62 20     0170
                                 2F     014C
```

```
                                        017C   257 FMT_SECTIONS:                                                       ;       [27]
6E 69 61 74 6E 6F 63 20 43 41 21 00'    017C   258         .ASCIC  "!AC contains the following sections:!/"            ;       [27]
77 6F 6C 6C 6F 66 20 65 68 74 20 73     0188
73 6E 6F 69 74 63 65 73 20 67 6E 69     0194
                        2F 21 3A        01A0
                              26        017C
                                        01A3   259
                                        01A3   260 FMT_TESTING:                                                        ;       [25]
23 21 20 3A 67 6E 69 74 73 65 54 00'    01A3   261         .ASCIC  \Testing: !#(7AS)!/\                               ;       [25]
                  2F 21 29 53 41 37 28  01AF
                                 12     01A3
                                        01B6   262
                                        01B6   263 FMT_MISMATCH:                                                       ;       [25]
76 65 72 20 73 69 68 54 20 3F 3F 00'    01B6   264         .ASCIC  "?? This rev of diagnostic will not run with this supervisor!/";[25]
74 73 6F 6E 67 61 69 64 20 66 6F 20     01C2
20 74 6F 6E 20 6C 6C 69 77 20 63 69     01CE
69 68 74 20 68 74 69 77 20 6E 75 72     01DA
72 6F 73 69 76 72 65 70 75 73 20 73     01E6
                              2F 21     01F2
                                 3D     01B6
                                        01F4   265
                                        01F4   266 FMT_NODEVICE:                                                       ;       [22]
76 65 64 20 6F 4E 20 3F 3F 2F 21 00'    01F4   267         .ASCIC  \!/?? No device support list in program.!/\        ;       [22]
20 74 72 6F 70 70 75 73 20 65 63 69     0200
67 6F 72 70 20 6E 69 20 74 73 69 6C     020C
                  2F 21 2E 6D 61 72     0218
                                 29     01F4
```

```
                     021E   269          .SBTTL  VRStart and VRReStart Routines
                 00000000   270          .PSECT  Code, Shr, Exe, NoWrt, Byte
                     0000   271 ;++
                     0000   272 ; FUNCTIONAL DESCRIPTION:
                     0000   273 ;
                     0000   274 ;       This subroutine is called from CLI to start the program.  It verifies the
                     0000   275 ;       command parameters and builds P-tables from the selected devices, and if
                     0000   276 ;       necessary, initializes the QIO database.
                     0000   277 ;
                     0000   278 ; CALLING SEQUENCE:
                     0000   279 ;
                     0000   280 ;       Subroutine call from the command line interpreter.
                     0000   281 ;
                     0000   282 ; INPUT PARAMETERS:
                     C000   283 ;
                     0000   284 ;       NONE
                     0000   285 ;
                     0000   286 ; IMPLICIT INPUTS:
                     0000   287 ;
                     0000   288 ;       R2                = BASE ADDRESS FOR CLI DATA BLOCK.
                     0000   289 ;       CLI$L_TEST(R2)    = FIRST TEST TO EXECUTE.
                     0000   290 ;       CLI$L_LAST(R2)    = LAST TEST TO EXECUTE.
                     0000   291 ;       CLI$L_PASS(R2)    = NUMBER OF PASSES TO RUN.
                     0000   292 ;       CLI$L_SUBT(R2)    = SUBTEST NUMBER FOR LOOP.
                     0000   293 ;       L$L_UNIT          = MAXIMUM NUMBER OF UNITS.
                     0000   294 ;
                     0000   295 ; OUTPUT PARAMETERS:    NONE
                     0000   296 ;
                     0000   297 ; IMPLICIT OUTPUTS:     NONE
                     0000   298 ;
                     0000   299 ; COMPLETION CODES:     NONE
                     0000   300 ;
                     0000   301 ; SIDE EFFECTS:
                     0000   302 ;
                     0000   303 ;       MAPFREE is called to initialize free memory Device for QIO may be initialize
                     0000   304 ;
                     0000   305 ; REGISTER USAGE:
                     0000   306 ;
                     0000   307 ;       R2 = BASE ADDRESS FOR CLI DATA BLOCK.
                     0000   308 ;       R3 = UNIT COUNT.
                     0000   309 ;       R4 = SECTION NAME ADDRESS LIST POINTER.
                     0000   310 ;       R5 = SECTION NAME LIST COUNTER.
                     0000   311 ;       R6 = SECTION NAME ASCII POINTER.
                     0000   312 ;       R7 = INPUT CHARACTER COUNT.
                     0000   313 ;       R8 = INPUT ASCII POINTER.
                     0000   314 ;       R9 = Equivalent name flag                                          [38]
                     0000   315 ;       R10= Ambiguous name flag                                           [38]
                     0000   316 ;--
```

```
                                0000   318 ;+
                                0000   319 ;       This is the entry point for the CLI call. Check to see if a program has [29]
                                0000   320 ;       been loaded.                                                            [29]
                                0000   321 ;-
                                0000   322
                                0000   323 VRSTART::
00000000'EF    01    E0         0000   324           BBS     #DS$V_LODFLG, -          ; CHECK LOAD FLAG.
                16               0007   325                     L^DS$GL_FLAGS, 10$    ;                                  [30]
                                 0008   326           $PRINT  #ds$k_type_command_err, -       ; Command error message      [26]
                                 0008   327                     #ds$k_printf, -       ;  use printf                      [26]
                                 0008   328                     L^FMT_START           ; "No program loaded"              [26]
    00000104'EF    9F            0008                         PUSHAB  L^FMT_START
          7E    01    9A         000E                         movzbl  #ds$k_printf, -(sp)
          7E    15    98         0011                         cvtbl   #ds$k_type_command_err, -(sp)
00000000'GF    03    FB         0014                         CALLS   #$$N+2, G^DSX$PRINT
             0321    31         001B   329           BRW     VRSTART_X                ; Exit routine                     [29]
                               001E   330
                               001E   331 10$:      Br_If_Apt       20$              ; APT sets this flag so don't check it [36]
0000FE00'EF    1F    E0         001E                BBS     #DSA$V_Apt, -                    ; If bit set, branch       [29]
                14               0025                      L^DSA$GL_Flags, 20$     ;                                  [29]
                1E    E1         0026   332           BBC     #DSA$V_LOAD_DEBUGGER,-   ;IF USER HAS TYPED "/DEBUG"        [36]
  0C 0000FE00'EF                 0028   333                     DSA$GL_FLAGS,20$       ;THEN                              [34]
     00000000'EF    16          002E   334           JSB     ACT_DEFAULT_DBG          ;   GET DEFAULT FILENAME FOR DEBUGGER [34]
     00000000'EF    16          0034   335           JSB     DSV$DEBUG                ;   LOAD AND START THE DEBUGGER      [34]
     40000000 8F    CA          003A   336 20$:      BICL2   #DSA$M_LOAD_DEBUGGER,-  ; Set up for new /DEBUG command      [36]
     0000FE00'EF                0040   337                     L^DSA$GL_FLAGS         ;                                  [36]
             02FD    30         0045   338           BSBW    Check_QA_Bit             ; Set or clear QA DSA bit          [30]
                                0048   339           QA_MAIN Start_VRStart            ; Call QA$Main                     [30]
                16    DD         0048                PUSHL   #DSQA$K_Start_VRStart
00000000'9F    01    FB         004A                CALLS   #1, @#QA$MAIN
```

```
                                0051   341 ;+
                                0051   342 ;       Check to make sure the diagnostic and this Supervisor match.            [29]
                                0051   343 ;       Also initialize context and call QA$Main.                               [29]
                                0051   344 ;-
                                0051   345
                                0051   346 VRRESTART::
                                0051   347 START1:
                00000000'EF   16 0051   348         JSB     SCRIPT$FLUSH            ; Flush scripts if console cmd          [24]
    50  00000204 9F   08   02  EF 0057   349         extzv   #env$v_super, -         ;                                       [26]
                                0060   350                   #env$s_super, -         ;                                       [26]
                                0060   351                   @#l$l_environ, r0       ; Fetch Diagnostic version              [26]
                      01   50   D1 0060   352         cmpl    r0, #env$_super         ; Is it identical?                      [26]
                           16   13 0063   353         BEQL    10$                     ; Branch if valid
                                0065   354         $PRINT  #ds$k_type_start_err, - ; Error starting diagnostic             [26]
                                0065   355                   #ds$k_printf, -         ; .. use PRINTF                         [26]
                                0065   356                   L^FMT_MISMATCH          ; Inform of mismatch                    [26]
                00000186'EF   9F 0065                       PUSHAB  L^FMT_MISMATCH
                      7E   01   9A 006B                     movzbl  #ds$k_printf, -(sp)
                      7E   18   98 006E                     cvtbl   #ds$k_type_start_err, -(sp)
              00000000'GF   03   FB 0071                     CALLS   #$$N+2, G^DSX$PRINT
                         02C4   31 0078   357         BRW     VRSTART_X               ; And Exit
                                007B   358
                00000000'EF   16 007B   359 10$:     JSB     INIT_CONTEXT            ; Reset stacks and modes                [24]
                         02C1   30 0081   360         BSBW    Check_QA_Bit            ; Set or clear QA DSA bit. Note: the    [30]
                                0084   361                                           ; QA bit was cleared by the             [30]
                                0084   362                                           ; SCRIPT$FLUSH routine above.           [30]
                                0084   363         QA_MAIN Start_VRRestart         ; Call QA$Main                          [30]
                           12   DD 0084                     PUSHL   #DSQA$K_Start_VRRestart
              00000000'9F   01   FB 0086                     CALLS   #1, @#QA$MAIN
```

```
                                 008D    365 ;+
                                 008D    366 ;       Cleanup the previous diagnostic and QIO (if applicable).        [29]
                                 008D    367 ;-
                                 008D    368
                00000000'EF  16  008D    369         JSB     DS_CLEANUP              ; DO CLEANUP IF NECESSARY       [24]
                                 0093    370
          0000FE00'EF    1C  E0  0093    371         BBS     #DSA$V_USER, -
                             07  009A    372                 DSA$GL_FLAGS, 15$       ; SKIP QIO$CLEANUP IN USERMODE
          00000000'EF    00  FB  009B    373         CALLS   #0,QIO$CLEANUP          ; Remove traces of QIO database
```

```
                                00A2   375 ;+
                                00A2   376 ;       Check the requested section name.                                    [29]
                                00A2   377 ;-
                                00A2   378
        52  00000000'EF    DE   00A2   379 15$:   MOVAL   DS$GL_CLIBASE, R2        ; Adress the CLI vector.             [30]
            0000FE10'EF    D4   00A9   380         CLRL    DSA$GL_SECTNO           ; INIT SECTION TO "DEFAULT".
                  10 A2    D5   00AF   381         TSTL    CLI$Q_SECTION(R2)       ; CHECK FOR SECTION SWITCH.
                     03    12   00B2   382         BNEQ    18$                     ; If found, continue                 [26]
                   00A7    31   00B4   383         brw     50$                     ; Branch                             [26]
                                00B7   384
        54  00000250'EF    D0   00B7   385 18$:   MOVL    L$A_SECNAM, R4          ; GET SECTION LIST ADDRESS.
                55   84    D0   00BE   386         MOVL    (R4)+, R5               ; GET SECTION COUNT.
                                00C1   387
                     5A    D4   00C1   388         CLRL    R10                     ; Ambiguous name Flag                [38]
                56   84    D0   00C3   389 20$:   MOVL    (R4)+, R6               ; GET SECTION NAME ASCII POINTER.
                     59    D4   00C6   390         CLRL    R9                      ; Equivalent name Flag               [38]
             57   10 A2    D0   00C8   391         MOVL    CLI$Q_SECTION(R2), R7   ; GET REQUESTED SECTION ASCII CNT.
             58   14 A2    D0   00CC   392         MOVL    CLI$Q_SECTION+4(R2), R8 ; GET STRING POINTER.
                57   66    91   00D0   393         CMPB    (R6), R7                ; CHECK LENGTHS.                     [38]
                     57    19   00D3   394         BLSS    40$                     ; BR IF INPUT TOO LONG.
                57   86    91   00D5   395         CMPB    (R6)+,R7                ; Characters Equivalent              [38]
                     02    12   00D8   396         BNEQ    30$                     ; No                                 [38]
                     59    D6   00DA   397         INCL    R9                      ; Yes                                [38]
                                00DC   398
                88   86    91   00DC   399 30$:   CMPB    (R6)+, (R8)+            ; COMPARE CHARACTERS.
                     4B    12   00DF   400         BNEQ    40$                     ; BR IF MISMATCH.
                   F8 57    F5   00E1   401         SOBGTR  R7, 30$                 ; COUNT CHARACTERS.
0000FE10'EF     00000250'FF    55   C3   00E4   402         SUBL3   R5, @L$A_SECNAM, -      ; COMPUTE SECTION NUMBER.
                                00F0   403                   DSA$GL_SECTNO
                     59    D5   00F0   404         TSTL    R9                      ; Equivalent name                    [38]
                     07    13   00F2   405         BEQL    32$                     ; No                                 [38]
             3B 62   16    E3   00F4   406         BBCS    #CLI$V_VALSEC, -        ; SET  VALID SECTION FLG.
                                00F8   407                   CLI$L_FLAGS(R2), 42$ ;                               [38]
                                00F8   408                                           ; Set Valid Section Flag        [38]
                   0038    31   00F8   409         BRW     42$                     ; All Done                           [38]
                                00FB   410 32$:                                      ;                               [38]
             2D 62   16    E3   00FB   411         BBCS    #CLI$V_VALSEC, -        ; SET AND CHECK VALID SECTION FLG.
                                00FF   412                   CLI$L_FLAGS(R2), 40$
                5A   01    D0   00FF   413         MOVL    #1, R10                 ; Indicate Ambiguous name            [38]
                   0027    31   0102   414         BRW     40$                     ; Continue with list                 [38]
                                0105   415 35$:                                      ;                               [38]
                                0105   416         $PRINT  #-ds$k_type_command_err, -       ; command error (sticky)    [26]
                                0105   417                   #ds$k_printf, -       ; .. use PRINTF                    [26]
                                0105   418                   L^FMT_AMBSEC          ; "?? Ambiguous section name."     [26]
            0000014C'EF    9F   0105                   PUSHAB  L^FMT_AMBSEC
               7E   01    9A   010B                   movzbl  #ds$k_printf, -(sp)
             7E   EB 8F    98   010E                   cvtbl   #-ds$k_type_commano_err, -(sp)
         00000000'GF    03    FB   0112                   CALLS   #$$N+2, G^DSX$PRINT
        50   00000250 9F    D0   0119   419         MOVL    @#L$A_SECNAM,R0         ; Point to section names
                   023F    30   0120   420         BSBW    PRLIST                  ; Print list
            00000000'EF    94   0123   421         clrb    L^ds$gb_typecode        ; Clear type code after done        [26]
                   0213    31   0129   422         BRW     VRSTART_X
```

```
                94 55   F5   012C   424 40$:    SOBGTR  R5, 20$                 ; COUNT SECTIONS.
                   5A   D5   012F   425          TSTL    R10                     ; Ambiguous Name found?                 [38]
                   D2   12   0131   426          BNEQ    35$                     ; Yes                                   [38]
                                0133   427 42$:                                  ;                                       [38]
             27 62   16   E0   0133   428          BBS     #CLI$V_VALSEC, -        ; CHECK FOR A VALID SECTION.
                                0137   429            CLI$L_FLAGS(R2), 50$
                                0137   430          $PRINT  #-ds$k_type_command_err, -      ; Command error (sticky)  [26]
                                0137   431            #ds$k_printf, -           ; .. use PRINTF                         [26]
                                0137   432            L^F[illegible]SEC         ; "?? Illegal section name."            [26]
        0000011E'EF   9F   0137                       PUSHA[illegible]MT_ILLSEC
            7E   01   9A   013D                       movzbl  #ds$k_printf, -(sp)
          7E   EB 8F   98   0140                       cvtbl   #-ds$k_type_command_err, -(sp)
    00000000'GF   03   FB   0144                       CALLS   #$$N+2, G^DSX$PRINT
50  00000250 9F   D0   014B   433          MOVL    @#L$A_SECNAM,R0             ; Point to section names
                020D   30   0152   434          BSBW    PRLIST                  ; Print section names
        00000000'EF   94   0155   435          clrb    L^ds$gb_typecode        ; Clear type code after done           [26]
                01E1   31   015B   436          BRW     VRSTART_X               ; Exit                                  [27]
```

```
                        015E   438 ;+
                        015E   439 ;       The section name checked out. Set the default flags.                    [29]
                        015E   440 ;-
                        015E   441
                        015E   442 50$:
                     CA 015E   443          BICL2   # DS$M_DONFLG ! -          ; CLEAR CLEANUP DONE
                        015F   444                    DS$M_STRFLG ! -          ; AND PROGRAM STARTED
                        015F   445                    DS$M_HLTFLG ! -          ; AND HALTED ON ERROR
                        015F   446                    DS$M_CTRLC  ! -          ; AND STOPPED BY ^C
                        015F   447                    DS$M_BRKPT  ! -          ; AND STOPPED BY BREAKPOINT
                        015F   448                    DS$M_EXCEPT   -          ; AND STOPPED BY EXCEPTION
00000000'EF   00082D0D 8F       015F   449                  , L^DS$GL_FLAGS          ; IN THE GLOBAL FLAGS                  [24]
```

```
                                0169   451 ;+
                                0169   452 ;       Check the requested test numbers (i.e. TEST:x:y).                [29]
                                0169   453 ;-
                                0169   454
      50   00000254'FF    D0  0169   455          MOVL    @L$A_TSTCNT, R0          ; GET ACTUAL TEST COUNT.
      00000000'EF    50  D0  0170   456          MOVL    R0,L^DS$GL_NUMTEST       ; SET HIGHEST TEST NUMBER       [24]
                     51  D4  0177   457          CLRL    R1                       ; Address page zero
0000FE08'EF    2C A2  D0  0179   458          MOVL    CLI$L_PASS(R2), -        ; PASS COUNT
                                0181   459            DSA$GL_PASSES
00000000'EF    20 A2  D0  0181   460          MOVL    CLI$L_TEST(R2), -        ; FIRST TEST NUMBER             [24]
                                0189   461            L^DS$GL_FSTTEST
                     43  19  0189   462          BLSS    95$                      ; MUST BE POSITIVE
                     06  12  018B   463          BNEQ    80$                      ; SKIP IF NOT BLANK.
            00000000'EF  D6  018D   464          INCL    L^DS$GL_FSTTEST          ; START AT TEST 1.              [24]
                                0193   465
00000000'EF    24 A2  D0  0193   466 80$:   MOVL    CLI$L_LAST(R2), -        ; LAST TEST NUMBER.             [24]
                                019B   467            L^DS$GL_LSTTEST          ;                               [24]
                     31  19  019B   468          BLSS    95$                      ; MUST BE POSITIVE
                     0B  12  019D   469          BNEQ    90$                      ; SKIP IF NOT 0.
00000000'EF  00000000'EF    D0  019F   470          MOVL    L^DS$GL_NUMTEST, -       ; DEFAULT LAST TEST.            [24]
                                01AA   471            L^DS$GL_LSTTEST          ;                               [24]
                                01AA   472
00000000'EF  00000000'EF    D1  01AA   473 90$:   CMPL    L^DS$GL_FSTTEST, -       ; FIRST TEST NUMBER IN RANGE?   [24]
                                01B5   474            L^DS$GL_NUMTEST          ;                               [24]
                     17  14  01B5   475          BGTR    95$                      ; NO, ERROR AND EXIT TO COMMAND
00000000'EF  00000000'EF    D1  01B7   476          CMPL    L^DS$GL_LSTTEST, -       ; LAST TEST NUMBER IN RANGE?    [24]
                                01C2   477            L^DS$GL_NUMTEST          ;                               [24]
                     0A  14  01C2   478          BGTR    95$                      ; NO, ERROR AND EXIT TO COMMAND
                                01C4   479
00000000'EF    28 A2  D0  01C4   480          MOVL    CLI$L_SUBT(R2), -        ; SUBTEST NUMBER.               [24]
                                01CC   481            L^DS$GL_SUBTEST          ;                               [24]
                     1C  11  01CC   482          BRB     100$                     ; CONTINUE
                                01CE   483
                                01CE   484 95$:   $PRINT  #ds$k_type_command_err, -        ; Command error      [26]
                                01CE   485            #ds$k_printf, -          ; .. use PRINTF                 [26]
                                01CE   486            L^FMT_TESTRANGE, -       ; .. format                     [26]
                                01CE   487            L^DS$GL_NUMTEST          ; .. error in test numbers      [26]
          00000000'EF  DD  01CE                PUSHL   L^DS$GL_NUMTEST
          000000A0'EF  9F  01D4                PUSHAB  L^FMT_TESTRANGE
               7E  01  9A  01DA                movzbl  #ds$k_printf, -(sp)
               7E  15  98  01DD                cvtbl   #ds$k_type_command_err, -(sp)
      00000000'GF  04  FB  01E0                CALLS   #$$N+2, G^DSX$PRINT
                0155  31  01E7   488          BRW     VRSTART_X                ; EXIT
```

```
                          01EA   490 ;+
                          01EA   491 ;       Print the diagnostic header lines, except under special circumstances.  [29]
                          01EA   492 ;-
                          01EA   493
                          01EA   494 100$:   QA_MAIN Start_Before_Header     ; Call QA$Main                          [30]
                    17    DD     01EA       PUSHL   #DSQA$K_Start_Before_Header
          00000000'9F   01    FB   01EC       CALLS   #1, @#QA$MAIN
                          01F3   495
          0000FE00'EF   0F    E1   01F3   496         BBC     #DSA$V_QA, -              ; If not running QA, branch.  [32]
                        1A         01FA   497           L^DSA$GL_FLAGS, 104$  ;                                      [32]
                        03    E0   01FB   498         BBS     #QA$K_LOOP_ON_SUBTEST,- ; If the Loop on Subtest check is [32]
                00000000'EF         01FD   499           L^QA$AOB_CHECK_STATE,-; executing, branch.                  [32]
                        0A          0202   500           103$                  ;                                     [32]
                        05    E0   0203   501         BBS     #QA$K_ERROR_PHASE_ONE,- ; If the Error Phase One check is [32]
                00000000'EF         0205   502           L^QA$AOB_CHECK_STATE,-; executing, branch.                  [32]
                        02          020A   503           103$                  ;                                     [32]
                        08    11   020B   504         BRB     104$                      ; Otherwise, branch and print header.  [32]
                          020D   505
          00000000'EF   06    E0   020D   506 103$:   BBS     #QA$K_NO_HEADER, -        ; If header should not be printed,  [32]
                        27         0214   507           L^QA$AOB_FLAGS, -       ; then branch around the print.       [32]
                          0215   508           105$                  ;                                             [32]
                          0215   509
                          0215   510 104$:   $PRINT  #ds$k_type_program_start, -     ; Start program               [32]
                          0215   511           #ds$k_printf, -           ; .. Use PRINTF                           [26]
                          0215   512           L^FMT_PROG, -             ; .. Format string                        [26]
                          0215   513           L$A_NAME(R1), -           ; .. Program name.                        [26]
                          0215   514           L$L_REV(R1), -            ; .. revision                             [26]
                          0215   515           L$L_UPDATE(R1), -         ; .. update                               [26]
                          0215   516           @l$a_tstcnt, -            ; .. number of tests                      [26]
                          0215   517           #0                        ; .. current time                         [26]
                    00    DD   0215                   PUSHL   #0
          00000254'FF   DD   0217                   PUSHL   @l$a_tstcnt
              0210 C1   DD   021D                   PUSHL   L$L_UPDATE(R1)
              020C C1   DD   0221                   PUSHL   L$L_REV(R1)
              0208 C1   DD   0225                   PUSHL   L$A_NAME(R1)
          00000006'EF   9F   0229                   PUSHAB  L^FMT_PROG
              7E    01   9A   022F                   movzbl  #ds$k_printf, -(sp)
              7E    0F   98   0232                   cvtbl   #ds$k_type_program_start, -(sp)
          00000000'GF   08   FB   0235                   CALLS   #$$N+2, G^DSX$PRINT
                          023C   518
                          023C   519 105$:                                    ;                                     [32]
```

```
                                 023C   521 ;+
                                 023C   522 ;       Make Ptables for all the units. Print what this diagnostic is testing.  [29]
                                 023C   523 ;-
                                 023C   524
                0000FE0C'EF   D4  023C   525          CLRL    DSA$GL_UNITS            ; Initialize unit counter
                00000220'EF   D5  0242   526          TSTL    L$L_UNIT                ; Does program test units?
                          03  12  0248   527          BNEQ    107$                    ; Yes, go generate Ptables for them.    [32]
                        0083  31  024A   528          BRW     170$                    ; No, skip unit dependent setup.        [32]
                                 024D   529
        00000000'EF     00    FB  024D   530 107$:    CALLS   #0,DSP$GEN_PTABLES      ; Make P-tables for selected units
                                 0254   531
    54  0000FE0C'EF     01    C3  0254   532          SUBL3   #1,DSA$GL_UNITS,R4      ; Highest unit number
                          27  18  025C   533          BGEQ    110$                    ; Branch if one found                   [29]
                                 025E   534
                                 C25E   535          $PRINT  #-ds$k_type_start_err, -; Error starting                        [26]
                                 025E   536                    #ds$k_printf, -       ; .. use PRINTF                         [26]
                                 025E   537                    L^FMT_NOUNITS         ; .. no units to test text              [26]
                000000CB'EF     9F  025E                      PUSHAB  L^FMT_NOUNITS
                       7E   01  9A  0264                      movzbl  #ds$k_printf, -(sp)
                     7E   E8 8F  98  0267                      cvtbl   #-ds$k_type_start_err, -(sp)
        00000000'GF     03    FB  026D                      CALLS   #$$N+2, G^DSX$PRINT
    50  0000021C 9F           D0  0272   538          MOVL    @#L$A_DEVP,R0           ; Get address of DEVTYP list
                        00E6  30  0279   539          BSBW    PRLIST                  ; Print device types                    [29]
                00000000'EF   94  027C   540          clrb    L^ds$gb_typecode        ; Clear sticky type code                [26]
                        00BA  31  0282   541          BRW     VRSTART_X               ; RETURN TO COMMAND MODE
```

```
                                0285    543 ;+
                                0285    544 ; If QA is running, AND if either the Loop on Subtest check or the          [32]
                                0285    545 ; Error Phase One check is executing, do not print the "Testing: ..."       [32]
                                0285    546 ; message unless the No_Header bit is False. This stops the message        [32]
                                0285    547 ; being printed too many times.                                            [32]
                                0285    548 ;-
                                0285    549
 0000FE00'EF    0F    E1       0285    550 110$:   BBC     #DSA$V_QA, -              ; If not running QA, branch.       [32]
                      1A        028C    551           L^DSA$GL_FLAGS, 120$  ;                                          [32]
                      03    E0  028D    552           BBS     #QA$K_LOOP_ON_SUBTEST,- ; If the Loop on Subtest check is  [32]
          00000000'EF           028F    553             L^QA$AOB_CHECK_STATE,-; executing, branch.                     [32]
                      0A        0294    554             115$                  ;                                          [32]
                      05    E0  0295    555           BBS     #QA$K_ERROR_PHASE_ONE,- ; If the Error Phase One check is  [32]
          00000000'EF           0297    556             L^QA$AOB_CHECK_STATE,-; executing, branch.                     [32]
                      02        029C    557             115$                  ;                                          [32]
                      08    11  029D    558           BRB     120$                    ; Otherwise, branch and print message. [32]
                                029F    559
 00000000'EF    06    E0       029F    560 115$:   BBS     #QA$K_NO_HEADER, -        ; If header should not be printed, [32]
                      29        02A6    561             L^QA$AOB_FLAGS, -     ; then branch around the print.            [32]
                                02A7    562             130$                  ;                                          [32]
                                02A7    563
                                02A7    564 120$:   $DS_GPHARD_S    R4,-(SP)        ; Get P-table address              [32]
                      7E    DF  02A7                PUSHAL  -(SP)
                      54    DD  02A9                PUSHL   R4
 00000000'9F          02    FB  02AB                CALLS   #2, @#DS$GPHARD
                   F2 54    F4  02B2    565         SOBGEQ  R4, 120$                ; Decrement and branch if more     [32]
                                02B5    566
      0000FE0C'EF       DD      02B5    567         PUSHL   DSA$GL_UNITS            ; Repeat count for !#(AD)
                50  6E    04    C1      02BB    568 ADDL3   #4,(SP),R0              ; Total parameter count            [26]
                 000001A3'EF    9F      C2BF    569 PUSHAB  L^FMT_TESTING           ; ASCIC EDIT STRING                [24]
                          01    DD      02C5    570 pushl   #ds$k_printf            ; Use printf                       [26]
                          17    DD      02C7    571 pushl   #ds$k_type_program_info ; Program info type code          [26]
 00000000'GF              50    FB      02C9    572 CALLS   R0, G^DSX$PRINT        ; TYPE IT                          [26]
                                02D0    573
                                02D0    574 130$:                               ;                                  [32]
```

```
                                02D0    576 ;+
                                02D0    577 ;       Add to the QIO database.        ;                                   [29]
                                02D0    578 ;-
                                02D0    579
          0000FE00'EF    1C    E0  02D0    580 170$:   BBS     #DSA$V_USER, -          ; BRANCH IF USER MODE
                           54        02D7    581                   DSA$GL_FLAGS, 180$
          00000000'EF    00    FB  02D8    582         CALLS   #0,L^MAPFREE            ; Map free memory                   [24]
          0000FE00'EF    1A    E1  02DF    583         BBC     #DSA$V_DEBUG, -         ; IF DEBUGGER IS RESIDENT           [34]
                           06        02E6    584                 DSA$GL_FLAGS,172$   ; THEN                              [34]
               00000000'EF    16  02E7    585         JSB     MAP_DEBUGGER            ;    MAP DEBUGGER'S MEMORY SPACE    [34]
                                02ED    586 172$:                                   ;                                   [34]
                                02ED    587         $DS_CLRVEC_S #SCB$L_TIMER       ; Reset timer vector
                   000000C0 8F  DD  02ED                       PUSHL   #SCB$L_TIMER
          00000000'9F    01    FB  02F3                       CALLS   #1, @#DS$CLRVEC
                                02FA    588
    02  00000204'EF    02    00  ED  02FA    589         CMPZV   #0,#2,L$L_ENVIRON, -
                                0303    590                 #SEP_FUNCTIONAL         ; System and Functional?
                           27    12  0303    591         BNEQ    180$                ; Branch if should not add unit to QIO
                           54    D4  0305    592         CLRL    R4                  ; Start with unit zero
                                0307    593
                                0307    594 175$:   $DS_GPHARD_S    R4,-(SP)       ; Get P-table address
                           7E    DF  0307                       PUSHAL  -(SP)
                           54    DD  0309                       PUSHL   R4
          00000000'9F    02    FB  030B                       CALLS   #2, @#DS$GPHARD
                     51    8E    D0  0312    595         MOVL    (SP)+,R1            ; P-table address
                   14 50    E9  0315    596         BLBC    R0,180$             ; Exit if error
                           51    DD  0318    597         PUSHL   R1                  ; Push for addunit
          00000000'EF    01    FB  031A    598         CALLS   #1,QIO$ADDUNIT      ; Add this unit to QIO database
                     1B 50    E9  0321    599         BLBC    R0,VRSTART_X        ; Return to CLI if errors
        DB 54    0000FE0C'EF    F2  0324    600         AOBLSS  DSA$GL_UNITS, R4, 175$  ; Count and continue if more units
```

```
                          032C   602 ;+
                          032C   603 ;       Reinitialize the stacks, set the started flag, dispatch the          [29]
                          032C   604 ;       tests, and go back to command mode. Note: the DISPATCH routine will  [29]
                          032C   605 ;       return to here only if the EndPass routine is NOT done.              [29]
                          032C   606 ;       Just before call to DISPATCH, do a $SETIMR if a /TIME switch was given [40]
                          032C   607 ;       on the START or RUN command. Since INIT_CONTEXT inits all registers  [40]
                          032C   608 ;       and enables ASTs, we do the $SETIMR after the call to INIT_CONTEXT   [40]
                          032C   609 ;       but can't use any regs after the call to INIT_CONTEXT. So, we test   [40]
                          032C   610 ;       CLI$Q_TIME before the call to INIT_CONTEXT, and clear BIN_TIME if no  [40]
                          032C   611 ;       /TIME was given. Then, after INIT_CONTEXT, test BIN_TIME to determine [40]
                          032C   612 ;       if we should $SETIMR. Load up BEGIN_TIME with the absolute time at   [40]
                          032C   613 ;       which we did the SETIMR, in case the user does a ^C and a CONTINUE   [40]
                          032C   614 ;       we can figure out how much more to time.                             [40]
                          032C   615 ;-
                          032C   616
                          032C   617 ;180$:  TSTL    CLI$Q_TIME(R2)          ; Any /TIME given ? Check CLI$Q_TIME  [40]
                          032C   618 ;                                         ; because it is initialized on every  [40]
                          032C   619 ;                                         ; command, BIN_TIME is not.           [40]
                          032C   620 ;       BNEQ    185$                    ; Branch if TIME was given            [40]
                          032C   621 ;       CLRQ    BIN_TIME                ; No TIME, init BIN_TIME              [40]
                          032C   622 ;
00000000'EF   16          032C   623 180$:  JSB     INIT_CONTEXT            ; REINITIALIZE STACKS ETC.            [24]
00000000'EF   04   C8     0332   624        BISL    #DS$M_STRFLG, -         ; SET PROGRAM STARTED                 [24]
                          0339   625                  L^DS$GL_FLAGS          ;                                     [24]
                          0339   626 ;       MOVQ    L^BIN_TIME,L^BIN_TIME   ; Any /TIME given? (test for zero)    [40]
                          0339   627 ;       BEQL    190$                    ; Branch if not                       [40]
                          0339   628 ;
                          0339   629 ;       $GETTIM_S TIMADR=BEGIN_TIME      ; Store current time, in case ^C, CONT [40]
                          0339   630 ;       $SETIMR_S EFN=#0, -              ; Supervisor efn                       [40]
                          0339   631 ;                 DAYTIM=BIN_TIME, -     ; Address of quad with 64-bit time     [40]
                          0339   632 ;                 ASTADR=DSI$TIME_AST, - ; AST to DSI$TIME_AST                  [40]
                          0339   633 ;                 REQIDT=#-2             ; ID for /TIME $SETIMR                 [40]
                          0339   634
00000000'EF   16          0339   635 190$:  JSB     DISPATCH                ; CALL PROGRAM                        [24]
                          033F   636
```

```
                                 033F    638 ;+
                                 033F    639 ;       End of the VRSTART routine. Return indirectly to the CLI routine.       [29]
                                 033F    640 ;-
                                 033F    641
                                 033F    642 VRSTART_X:
                00000000'EF   17 033F    643         JMP     BEGIN                   ; Return to command decoder                 [24]
```

```
                                0345   645 ;+
                                0345   646 ;       Routine to turn the QA flag on or off depending on whether              [29]
                                0345   647 ;       or not the CLI QA flag was set. This routine destroys R0.               [30]
                                0345   648 ;       Usage:                                                                  [30]
                                0345   649 ;               BSBB    Check_QA_Bit                                            [30]
                                0345   650 ;               BSBW    Check_QA_Bit                                            [30]
                                0345   651 ;               JSB     Check_QA_Bit                                            [30]
                                0345   652 ;-
                                0345   653
                                0345   654 Check_QA_Bit:
50   00000000'EF    DE          0345   655         MOVAL   DS$GL_CLIBASE, R0       ; Readdress CLI command area           [30]
      09 60   07    E1          034C   656         BBC     #CLI$V_QA, -            ; If bit clear, go clear DSA bit       [23]
                                0350   657           CLI$L_FLAGS (R0), 20$ ;                                             [30]
0000FE00'EF    0F    E3          0350   658         BBCS    #DSA$V_QA, -            ; Set DSA QA bit                       [23]
               00                0357   659           DSA$GL_FLAGS, 10$     ;                                             [30]
                                0358   660
                          05    0358   661 10$:    RSB                             ; Return to caller                     [30]
                                0359   662
0000FE00'EF    0F    E5          0359   663 20$:    BBCC    #DSA$V_QA, -            ; Clear DSA QA bit                     [30]
               00                0360   664           DSA$GL_FLAGS, 30$     ;                                             [30]
                                0361   665                                 ;                                             [23]
                                0361   666
                          05    0361   667 30$:    RSB                             ; Return to caller                     [30]
```

```
                                0362   669 ;+
                                0362   670 ; This routine is used to print a list of ascic NAMES. R0= address of list.
                                0362   671 ; Usage:
                                0362   672 ;       JSB     PRLIST
                                0362   673 ;       BSBW    PRLIST
                                0362   674 ;       BSBB    PRLIST
                                0362   675 ;-
                                0362   676
                                0362   677 PRLIST:
            51    60    D0      0362   678         MOVL    (R0),R1                 ; Get number of entries in list
         7E   2F21  8F    B0    0365   679         MOVW    #^A"!/",-(SP)           ; Create edit string starting with "!/"
                 05    11      036A   680         BRB     20$                     ; Skip first ","
         7E   202C  8F    B0    036C   681 10$:    MOVW    #^A", ",-(SP)           ; append ", "
         7E   4341  8F    B0    0371   682 20$:    MOVW    #^A"AC",-(SP)           ; Append "AC"
              7E    21    90    0376   683         MOVB    #^A"!",-(SP)            ; Append "!"
                F0  51    F5    0379   684         SOBGTR  R1,10$                  ; one for each parameter in list
         51   60    05    C5    037C   685         MULL3   #5,(R0),R1              ; Get count back
              7E    51    90    0380   686         MOVB    R1,-(SP)                ; Make edit string ASCIC
              51    60    D0    0383   687         MOVL    (R0),R1                 ; Get count back
                 6041    DD    0386   688 30$:    PUSHL   (R0)[R1]                ; Push address of argument
                FA  51    F5    0389   689         SOBGTR  R1,30$                  ; Each address now on stack
              51    60    D0    038C   690         MOVL    (R0),R1                 ; Get count back
                 6E41    DF    038F   691         PUSHAL  (SP)[R1]                ; Address of ASCIC string
              51    03    C0    0392   692         ADDL2   #3,R1                   ; Total count of print args             [33]
              7E    01    9A    0395   693         movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                            [33]
              7E    25    98    0398   694         cvtbl   #ds$k_type_start_list,-(sp)     ; Code                          [33]
00000000'GF   51    FB    039B   695         CALLS   R1, G^DSX$PRINT                 ; Print the text                        [33]
              50    8E    9A    03A2   696         MOVZBL  (SP)+,R0                ; Get length of edit string
              5E    50    C0    03A5   697         ADDL    R0,SP                   ; Remove edit string from stack
                          05    03A8   698         RSB                             ; Return
```

```
                                        .SBTTL  VRAbort and DSX$Abort Routines
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       ABORTS CURRENT USER PROGRAM AFTER ERROR HALT, ^C, BREAKPOINT, OR
                                ;       EXCEPTION, OR /TIME HAS EXPIRED.                                        [40]
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBW    VRABORT
                                ; -or-  CALLG   (SP),DS$ABORT
                                ; -or-  from DSI$TIME_AST, fix up the argument list to return to VRABORT        [40]
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       NONE
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       (AP)    - USER REGISTER LIST.
                                ;       DS$GL_FLAGS:
                                ;               DS$V_BRKPT      - BREAKPOINT FLAG.
                                ;               DS$V_EXCEPT     - EXCEPTION FLAG.
                                ;               DS$V_HLTFLG     - ERROR HALT FLAG.
                                ;               DS$V_CMDFLG     - COMMAND MODE FLAG.
                                ;               DS$V_DONFLG     - PROGRAM COMPLETE FLAG.
                                ;
                                ; OUTPUT PARAMETERS:    NONE
                                ;
                                ; IMPLICIT OUTPUTS:     NONE
                                ;
                                ; COMPLETION CODES:     NONE
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       ALL USER PROGRAM CONTEXT CLEARED.
                                ;--
```

```
                          0000  03A9   738 .ENTRY  DSX$ABORT,^M<>
                                03AB   739
          0000FE00'EF     0F    E1  03AB   740          BBC     #DSA$V_QA, -                ; If not running QA, branch.              [32]
                            16        03B2   741            L^DSA$GL_FLAGS, 10$       ;                                         [32]
                            05    E1  03B3   742          BBC     #QA$K_ERROR_PHASE_ONE, -; If not running the Error Phase One     [32]
                                03B5   743            L^QA$AOB_CHECK_STATE,-; check, branch.                                  [32]
                       0E 00000000'EF       03B5   744            10$             ;                                         [32]
                                CA  03BB   745          BICL2   #DS$M_HLTFLG ! -            ; Clear halt on error.                    [32]
                                03BC   746            DS$M_BRKPT  ! -           ; Clear "breakpoint" flag.                [32]
                                03BC   747            DS$M_EXCEPT   -           ; Clear "exception" flag.                 [32]
    00000000'EF   00080808 8F       03BC   748          ,L^DS$GL_FLAGS            ;                                         [32]
                        FC88    31  03C6   749          BRW     VRRESTART                   ; Go back and Restart the diagnostic.     [32]
                                03C9   750
                 50   10 AD    D0  03C9   751 10$:     MOVL    16(FP),R0                   ; Get callers PC                          [32]
                            04    11  03CD   752          Brb     Is_It_User_PC               ; Is this really a user PC?               [37]
                                03CF   753
                                03CF   754 ;DSI$ABORT:                                  ; Entry point for /TIME ABORT             [40]
                                03CF   755 ;        BISL2   #DS$M_SETIMR, -             ; Set /TIME flag                          [40]
                                03CF   756 ;                L^DS$GL_FLAGS               ;                                         [40]
                                03CF   757 ;        BRB     IS_IT_USER_PC               ; Join common code                        [40]
                                03CF   758
                                03CF   759 VRABORT::
                 50   3C AC    D0  03CF   760          MOVL    60(AP),R0                   ; Get PC from CLI args
                                03D3   761
                                03D3   762          ;+
                                03D3   763          ; Now, check to make sure this is a user's PC (i.e., less than            [37]
                                03D3   764          ; %X'10000'). If R0 is GTR 10000, then look for the user's PC on the      [37]
                                03D3   765          ; call frame stack.                                                       [37]
                                03D3   766          ;-
                                03D3   767
                                03D3   768 Is_It_User_PC:                               ; Is this really a user PC?               [37]
          00010000'EF     50    D1  03D3   769          CmpL    R0, ^X00010000              ; Compare the start of VDS code           [37]
                            07    1B  03DA   770          BlequU  VRAbort_Com                 ; If less than 10000, it's a user PC      [41]
          00000000'EF     00    FB  03DC   771          CallS   #0, L^DSR$Find_User_PC      ; Search on the stack for it              [37]
                                03E3   772
                                03E3   773 VRABORT_COM:
                                03E3   774          ;+
                                03E3   775          ; R0 now contains either a user PC or 0.                                  [37]
                                03E3   776          ;-
                            01  BB  03E3   777          PUSHR   #^MR0                       ; Save PC at error for PRINTF below
                                03E5   778
                00000000'EF     16  03E5   779          JSB     SCRIPT$FLUSH                ; Flush scripts if console command       [24]
                                CA  03EB   780          BICL2   # DS$M_HLTFLG ! -           ; CLEAR HALT ON ERROR.
                                03EC   781            DS$M_BRKPT  ! -           ; CLEAR "BREAKPOINT" FLAG.
                                03EC   782            DS$M_EXCEPT   -           ; CLEAR "EXCEPTION" FLAG.
    00000000'EF   00080808 8F       03EC   783          , L^DS$GL_FLAGS           ;                                         [24]
          00000000'EF     02    E0  03F6   784          BBS     #DS$V_STRFLG, -             ; Started?                                [24]
                            03        03FD   785            L^DS$GL_FLAGS,5$          ;                                         [24]
                        0074    31  03FE   786          BRW     100$                        ; Branch if not started                   [40]
                00000000'EF     16  0401   787 5$:     JSB     DS_TESTID                   ; GET TEST/SUBTEST STRING.                [35]
                                0407   788
                            01  BA  0407   789          POPR    #^MR0                       ; Restore PC at error
                     50     00    D1  0409   790          CmpL    #0, R0                      ; Did we find a real user PC?             [37]
                            23    13  040C   791          Beql    15$                         ; Branch if we did not                    [37]
                                040E   792 ;        BBSC    #DS$V_SETIMR, -             ; ABORT or /TIME ran out?                 [40]
                                040E   793 ;                L^DS$GL_FLAGS,10$           ;                                         [40]
                                040E   794
```

```
                                040E   795          $PRINT  #ds$k_type_abort_program, -; Abort program message              [26]
                                040E   796                    #ds$k_printf, -       ; ... use PrintF                      [26]
                                040E   797                    L^FMT_ABORT, -        ; ... abort message                   [26]
                                040E   798                    DSA$GL_PASSNO, -      ; ... current pass number            [37]
                                040E   799                    #DS$GQ_TESTID, -      ; ... test id                        [37]
                                040E   800                    R0                    ; ... user PC                        [37]
                     50   DD    040E                         PUSHL   R0
          00000000'8F     DD    0410                         PUSHL   #DS$GQ_TESTID
          0000FE54'EF     DD    0416                         PUSHL   DSA$GL_PASSNO
          00000046'EF     9F    041C                         PUSHAB  L^FMT_ABORT
              7E   01     9A    0422                         movzbl  #ds$k_printf, -(sp)
              7E   14     98    0425                         cvtbl   #ds$k_type_abort_program, -(sp)
    00000000'GF    06     FB    0428                         CALLS   #$$N+2, G^DSX$PRINT
                   21     11    042F   801          Brb     25$                   ; Do rest of abort                   [37]
                                0431   802
                                0431   803 ;10$:    $PRINT  #DS$K_TYPE_ABORT_PROGRAM, -     ; CRD should never see this. [40]
                                0431   804 ;                  #DS$K_PRINTF, -       ; PRINTF                             [40]
                                0431   805 ;                  L^FMT_TIME_EXPIRED, - ; the message                        [40]
                                0431   806 ;                  DSA$GL_PASSNO, -      ; current pass number                [40]
                                0431   807 ;                  #DS$GQ_TESTID, -      ; test and subtest                   [40]
                                0431   808 ;                  R0
                                0431   809 ;          BRB     25$                   ; Do rest of /TIME abort             [40]
                                0431   810
                                0431   811 ;15$:    BBSC    #DS$V_SETIMR, -       ; ABORT or /TIME ran out?            [40]
                                0431   812 ;                  L^DS$GL_FLAGS,20$     ;                                    [40]
                                0431   813
                                0431   814 ;5$:     $Print  #DS$K_Type_Abort_Program, -; Abort program typecode          [37]
                                0431   815                    #DS$K_PrintF, -       ; ... use PrintF                     [37]
                                0431   816                    L^Fmt_Abort_No_PC, -  ; ... abort message                  [37]
                                0431   817                    DSA$GL_PassNo, -      ; ... the current pass number       [37]
                                0431   818                    #DS$GQ_TestId         ; ... test/subtest                   [37]
          0000000C'8F     DD    0431                         PUSHL   #DS$GQ_Testid
          0000FE54'EF     DD    0437                         PUSHL   DSA$GL_PassNo
          00000077'EF     9F    043D                         PUSHAB  L^Fmt_Abort_No_PC
              7E   01     9A    0443                         movzbl  #DS$K_PrintF, -(sp)
              7E   14     98    0446                         cvtbl   #DS$K_Type_Abort_Program, -(sp)
    00000000'GF    05     FB    0449                         CALLS   #$$N+2, G^DSX$PRINT
                   00     11    0450   819          BRB     25$                   ; Finish the ABORT                   [40]
                                0452   820
                                0452   821 ;20$:    $PRINT  #DS$K_TYPE_ABORT_PROGRAM, -     ; CRD should never see this. [40]
                                0452   822 ;                  #DS$K_PRINTF, -       ; PRINTF                             [40]
                                0452   823 ;                  L^FMT_TIME_EXPIRED_NO_PC, -     ; the message          [40]
                                0452   824 ;                  DSA$GL_PASSNO, -      ; current pass number                [40]
                                0452   825 ;                  #DS$GQ_TESTID         ; test and subtest                   [40]
                                0452   826
    0000FE00'EF     1F    E1    0452   827 25$:     Bbc     #DSA$V_Apt, -         ; Clean up if not in APT mode        [37]
                   08           0459   828                    DSA$GL_Flags, 50$     ;                                    [21]
          00000000'EF     04    E0    045A   829          Bbs     #DS$V_ErrFlg, -       ; Skip cleanup if APT mode           [21]
                   0C           0461   830                    DS$GL_Flags, 75$      ; ... and there was an error        [21]
                                0462   831
          00000C00'EF     16    0462   832 50$:     Jsb     Init_Context          ; Reset stacks and all               [24]
          00000000'EF     16    0468   833          Jsb     DS_Cleanup            ; Do user cleanup                    [24]
                                046E   834
          00000000'EF     04    CA    046E   835 75$:     BicL    #DS$M_StrFlg, -       ; Clear start flag                   [21]
                                0475   836                    DS$GL_Flags           ;                                    [21]
                                0475   837
    0000FE40'EF     06    D0    0475   838 100$:    MOVL    #APM$_ABTDON, -
```

```
                          047C   839                    DSA$GL_MSGTYP          ; YES- INDICATE ABORT DONE
                00000000'EF   17  047C   840      JMP  BEGIN                  ; TRANSFER TO SUPERVISOR CLEANUP.      [24]
```

```
0482   842          .SBTTL  VRSupport Routines
0482   843 ;++
0482   844 ; FUNCTIONAL DESCRIPTION:
0482   845 ;
0482   846 ;       This subroutine is called from CLI to show the devices a diagnostic
0482   847 ;       supports.  It will issue messages for no diagnostics loaded, no
0482   848 ;       devices supported.
0482   849 ;
0482   850 ; CALLING SEQUENCE:
0482   851 ;
0482   852 ;       SUBROUTINE CALL FROM THE COMMAND LINE INTERPRETER.
0482   853 ;
0482   854 ; INPUT PARAMETERS:
0482   855 ;
0482   856 ;       NONE
0482   857 ;
0482   858 ; IMPLICIT INPUTS:
0482   859 ;
0482   860 ;       NONE
0482   861 ;
0482   862 ; OUTPUT PARAMETERS:     NONE
0482   863 ;
0482   864 ; IMPLICIT OUTPUTS:      NONE
0482   865 ;
0482   866 ; COMPLETION CODES:      NONE
0482   867 ;
0482   868 ; SIDE EFFECTS:          NONE
0482   869 ;
0482   870 ;--
```

```
                                 0482   872 VRSUPPORT::                                    ;                                         [22]
         00000000'EF   01   E0   0482   873          BBS     #DS$V_LODFLG, -               ; Check load flag.                        [22]
                       14        0489   874                    L^DS$GL_FLAGS, 10$          ;                                         [22]
                                 048A   875 5$:      $PRINT  #ds$k_type_command_err,-      ; Command error                           [26]
                                 048A   876                     #ds$k_printf, -            ; .. use PRINTF                           [26]
                                 048A   877                     L^FMT_START                ;                                         [22]
            00000104'EF   9F     048A                        PUSHAB  L^FMT_START
                   7E   01   9A  0490                        movzbl  #ds$k_printf, -(sp)
                   7E   15   98  0493                        cvtbl   #ds$k_type_command_err, -(sp)
         00000000'GF   03   FB   0496                        CALLS   #$$N+2, G^DSX$PRINT
                            05   049D   878          RSB                                   ; Return to caller.                       [22]
                                 049E   879
   01    00000204 9F   08   02   ED  049E  880 10$:  cmpzv   #env$v_super, -               ; Make sure a diagnostic is really        [26]
                                 04A7   881                   #env$s_super, -              ; .. loaded, and has the correct          [26]
                                 04A7   882                   @#l$l_environ, -             ; .. version for this                     [26]
                                 04A7   883                   #env$_super                  ; .. Supervisor                           [26]
                       E1   12   04A7   884          bneq    5$                            ; If not, error                           [26]
         50     0000021C 9F  D0  04A9   885          MOVL    @#L$A_DEVP,R0                 ; Get address of DEVTYP list              [22]
                   51   60   D0  04B0   886          MOVL    (R0),R1                       ; Get value pointed to                    [22]
                       51   D5   04B3   887          TSTL    R1                            ; Check if device list exists             [22]
                       14   12   04B5   888          BNEQ    20$                           ; Brancn if list exists                   [22]
                                 04B7   889
                                 04B7   890          $PRINT  #ds$k_type_command_out,-      ; Command output (no devices)             [26]
                                 04B7   891                    #ds$k_printf, -             ; .. use PRINTF                           [26]
                                 04B7   892                    L^FMT_NODEVICE              ; "No device supported."                  [22]
            000001F4'EF   9F     04B7                        PUSHAB  L^FMT_NODEVICE
                   7E   01   9A  04BD                        movzbl  #ds$k_printf, -(sp)
                   7E   16   98  04C0                        cvtbl   #ds$k_type_command_out, -(sp)
         00000000'GF   03   FB   04C3                        CALLS   #$$N+2, G^DSX$PRINT
                            05   04CA   893          RSB                                   ; Return to caller                        [20]
                                 04CB   894
                       16   90   04CB   895 20$:     movb    #ds$k_type_command_out,-      ; Set command type code                   [26]
            00000000'EF          04CD   896                    L^ds$gb_typecode            ; .. in global                            [26]
                     FE8D   30   04D2   897          BSBW    PRLIST                        ; Print device types                      [20]
            00000000'EF   94     04D5   898          clrb    L^ds$gb_typecode              ; Clear type code                         [26]
                            05   04DB   899          RSB                                   ; Return to caller                        [20]
```

```
04DC    901          .SBTTL  VRShowSections Routine
04DC    902 ;++
04DC    903 ; FUNCTIONAL DESCRIPTION:
04DC    904 ;
04DC    905 ;       This subroutine is called from CLI to show the sections that
04DC    906 ;       a diagnostic supports.
04DC    907 ;
04DC    908 ; CALLING SEQUENCE:
04DC    909 ;
04DC    910 ;       JSB     VRSHOWSECTIONS
04DC    911 ;
04DC    912 ; INPUT PARAMETERS:
04DC    913 ;
04DC    914 ;       NONE
C4DC    915 ;
04DC    916 ; IMPLICIT INPUTS:
04DC    917 ;
04DC    918 ;       L$A_SECNAM - Pointer to list of section names
04DC    919 ;       L$A_NAME   - Pointer to diagnostic program name
04DC    920 ;
04DC    921 ; OUTPUT PARAMETERS:
04DC    922 ;
04DC    923 ;       NONE
04DC    924 ;
04DC    925 ; IMPLICIT OUTPUTS:
04DC    926 ;
04DC    927 ;       NONE
04DC    928 ;
04DC    929 ; COMPLETION CODES:
04DC    930 ;
04DC    931 ;       NONE
04DC    932 ;
04DC    933 ; SIDE EFFECTS:
04DC    934 ;
04DC    935 ;       NONE
04DC    936 ;
04DC    937 ;--
```

```
                                        04DC   939 VRSHOWSECTIONS::
                                        04DC   940
                00000000'EF   01   E0    04DC   941          BBS     #DS$V_LODFLG, -                         ; Check load flag.              [27]
                              14         04E3   942                    L^DS$GL_FLAGS, 10$                    ;                               [27]
                                        04E4   943 5$:       $PRINT  #ds$k_type_command_err, -               ; Command error                 [26]
                                        04E4   944                    #ds$k_printf, -                       ; .. use PRINTF                 [26]
                                        04E4   945                    L^FMT_START                           ; .. No diagnostic running.     [27]
                      00000104'EF   9F    04E4                       PUSHAB  L^FMT_START
                          7E   01   9A    04EA                       movzbl  #ds$k_printf, -(sp)
                          7E   15   98    04ED                       cvtbl   #ds$k_type_command_err, -(sp)
                00000000'GF   03   FB    04F0                       CALLS   #$$N+2, G^DSX$PRINT
                              05         04F7   946          RSB                                             ; Return to caller.             [27]
                                        04F8   947
    01  00000204 9F   08   02   ED    04F8   948 10$:      cmpzv   #env$v_super, -         ; Make sure a diagnostic is really      [26]
                                        0501   949                    #env$s_super, -       ; .. loaded, and has the correct        [26]
                                        0501   950                    @#l$l_environ, -      ; .. version for this                   [26]
                                        0501   951                    #env$_super           ; .. Supervisor                         [26]
                              E1   12   0501   952          bneq    5$                      ; If not, error                         [26]
                                        0503   953          $PRINT  #-ds$k_type_command_out, -              ; Command output (sticky)       [26]
                                        0503   954                    #ds$k_printf, -                       ; .. use PRINTF                 [26]
                                        0503   955                    L^FMT_SECTIONS, -                     ; .. format                     [26]
                                        0503   956                    L^L$A_NAME                            ; .. the sections message       [27]
                      00000208'EF   DD    0503                       PUSHL   L^L$A_NAME
                      0000017C'EF   9F    0509                       PUSHAB  L^FMT_SECTIONS
                          7E   01   9A    050F                       movzbl  #ds$k_printf, -(sp)
                        7E   EA   8F    0512                       cvtbl   #-ds$k_type_command_out, -(sp)
                00000000'GF   04   FB    0516                       CALLS   #$$N+2, G^DSX$PRINT
    50  00000250 9F   D0   051D   957          MOVL    @#L$A_SECNAM,R0                                 ; Point to section names        [27]
                    FE3B   30   0524   958          BSBW    PRLIST                                          ; Print list of sections        [27]
                00000000'EF   94   0527   959          clrb    L^ds$gb_typecode                        ; Clear sticky code             [26]
                              05   052D   960          RSB                                             ;                               [27]
                                   052E   961
```

```
                        963          .SBTTL  DSI$TIME_AST Routine
                        964 ;++
                        965 ; FUNCTIONAL DESCRIPTION:
                        966 ;
                        967 ;       This is the AST routine for the /TIME switch on the START command.      [40]
                        968 ;       Since we come here
                        969 ;       via an AST from a $SETIMR, the AP points to a parameter list which has
                        970 ;       the PC in the program or supervisor to return to. This routine modifies
                        971 ;       that PC to point to the DSI$ABORT routine. Note that the PC in the call
                        972 ;       frame cannot be changed, since it points to a VMS (or VDS) routine
                        973 ;       which cleans up after the AST. The value of R0 which is contained in the
                        974 ;       parameter list is replaced with the PC, so that DSI$ABORT can decide
                        975 ;       if the PC was in the program or not.
                        976 ;
                        977 ; CALLING SEQUENCE:
                        978 ;
                        979 ;       Called with a CALLG from system service which delivers ASTs.
                        980 ;
                        981 ; INPUT PARAMETERS:
                        982 ;
                        983 ;       NONE
                        984 ;
                        985 ; IMPLICIT INPUTS:
                        986 ;
                        987 ;       NONE
                        988 ;
                        989 ; OUTPUT PARAMETERS:    NONE
                        990 ;
                        991 ; IMPLICIT OUTPUTS:     Sets DS$V_SETIMR.
                        992 ;
                        993 ; COMPLETION CODES:     NONE
                        994 ;
                        995 ; SIDE EFFECTS:        Causes VDS to abort the program.
                        996 ;
                        997 ;--
                        998
                        999 ;.ENTRY DSI$TIME_AST,^M<>                 ;                                  [40]
                       1000 ;       MOVL    T6(AP),8(AP)               ; Move PC at which we were interrupted  [40]
                       1001 ;                                          ; to R0 in the AST arg list, so that  [40]
                       1002 ;                                          ; DSI$ABORT can print it out          [40]
                       1003 ;       MOVAB   DSI$ABORT,16(AP)           ; Abort the program by  replacing the  [40]
                       1004 ;                                          ; PC in  the AST argument list.       [40]
                       1005 ;       RET                                ; Back to AST deliverer.              [40]
                       1006
                       1007         .END
```

```
$$N                             =  00000002     D
$ER                             =  00000001     D
$MODULE                            00000000 R   D   02
ACT_DEFAULT_DBG                    ********     X   03
APC$_ABORT                      =  00000006     D
APC$_CONTINUE                   =  00000009     D
APC$_DUMP                       =  00000003     D
APC$_NOP                        =  00000000     D
APC$_START                      =  00000005     D
APC$_ZERO                       =  00000004     D
APM$_ABTDON                     =  00000006     D
APM$_DEVERR                     =  00000002     D
APM$_DONE                       =  0000000A     D
APM$_EXCEPT                     =  0000000B     D
APM$_HARDERR                    =  00000003     D
APM$_MORE                       =  00000008     D
APM$_NOMES                      =  00000000     D
APM$_PREPERR                    =  0000000C     D
APM$_PRGERR                     =  00000005     D
APM$_SOFTERR                    =  00000004     D
APM$_SPOOL                      =  00000009     D
APM$_SYSERR                     =  00000001     D
BEGIN                              ********     X   03
BIT...                          =  00000004     D
CEP_FUNCTIONAL                  =  00000000     D
CEP_REPAIR                      =  00000001     D
CF$L_AP                            00000008     D
CF$L_FP                            0000000C     D
CF$L_ONCOND                        00000000     D
CF$L_PC                            00000010     D
CF$L_REG                           00000014     D
CF$W_MASK                          00000006     D
CF$W_PSW                           00000004     D
CHECK_QA_BIT                       00000345 R   D   03
CLI$K_ALNUM                     =  00000087     D
CLI$K_ALPHA                     =  00000086     D
CLI$K_BIF                       =  00000083     D
CLI$K_BIFS                      =  00000094     D
CLI$K_BR                        =  00000082     D
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_CALL                      =  00000092     D
CLI$K_COMMA                     =  0000008E     D
CLI$K_DEC                       =  0000008A     D
CLI$K_EOL                       =  00000091     D
CLI$K_ERROR                     =  00000080     D
CLI$K_EXIT                      =  00000081     D
CLI$K_FILE                      =  00000095     D
CLI$K_HEX                       =  00000089     D
CLI$K_KEYWORD                   =  0000008C     D
CLI$K_NUM                       =  00000085     D
CLI$K_OCT                       =  00000088     D
CLI$K_RETURN                    =  00000093     D
CLI$K_SIZE                         00000444     D
CLI$K_SLASH                     =  00000090     D
CLI$K_SPACE                     =  00000084     D
CLI$K_STRING                    =  0000008B     D
CLI$K_SYMBOL                    =  0000008D     D
CLI$K_VALUE                     =  0000008F     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_LAST                         00000024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEST                         00000020     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       0000003C     D
CLI$V_ADAPTER                   =  00000018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BREAK                     =  0000000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
DISPATCH                           ********     X   03
DS$CLRVEC                          ********     X   03
DS$GB_TYPECODE                     ********     X   03
DS$GL_CLIBASE                      ********     X   03
DS$GL_FLAGS                        ********     X   03
DS$GL_FSTTEST                      ********     X   03
DS$GL_LSTTEST                      ********     X   03
DS$GL_NUMTEST                      ********     X   03
DS$GL_SUBTEST                      ********     X   03
```

```
DS$GPHARD                       ********   X   03
DS$GQ_TESTID                    ********   X   03
DS$K_BBC                        = 0000001D   G D
DS$K_BBCC                       = 00000021   G D
DS$K_BBCCI                      = 00000023   G D
DS$K_BBCS                       = 0000001F   G D
DS$K_BBS                        = 0000001C   G D
DS$K_BBSC                       = 00000020   G D
DS$K_BBSS                       = 0000001E   G D
DS$K_BBSSI                      = 00000022   G D
DS$K_BCC_M                      = 0000001B   G D
DS$K_BCS_M                      = 0000001A   G D
DS$K_BEQLU_B                    = 00000005   G D
DS$K_BEQLU_M                    = 00000011   G D
DS$K_BEQL_B                     = 00000004   G D
DS$K_BEQL_M                     = 00000010   G D
DS$K_BERROR                     = 00000026   G D
DS$K_BGEQU_B                    = 00000007   G D
DS$K_BGEQU_M                    = 00000013   G D
DS$K_BGEQ_B                     = 00000006   G D
DS$K_BGEQ_M                     = 00000012   G D
DS$K_BGTRU_B                    = 00000009   G D
DS$K_BGTRU_M                    = 00000015   G D
DS$K_BGTR_B                     = 00000008   G D
DS$K_BGTR_M                     = 00000014   G D
DS$K_BLBC                       = 00000025   G D
DS$K_BLBS                       = 00000024   G D
DS$K_BLEQU_B                    = 00000003   G D
DS$K_BLEQU_M                    = 0000000F   G D
DS$K_BLEQ_B                     = 00000002   G D
DS$K_BLEQ_M                     = 0000000E   G D
DS$K_BLSSU_B                    = 00000001   G D
DS$K_BLSSU_M                    = 0000000D   G D
DS$K_BLSS_B                     = 00000000   G D
DS$K_BLSS_M                     = 0000000C   G D
DS$K_BNEQU_B                    = 0000000B   G D
DS$K_BNEQU_M                    = 00000017   G D
DS$K_BNEQ_B                     = 0000000A   G D
DS$K_BNEQ_M                     = 00000016   G D
DS$K_BNERROR                    = 00000027   G D
DS$K_BVC_M                      = 00000019   G D
DS$K_BVS_M                      = 00000018   G D
DS$K_DS_STARTING_LOCATION       = 00010000     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
```

```
DS$V_BRKCLR                     =  0000000C     D
DS$V_BRKPT                      =  0000000B     D
DS$V_CHARFLG                    =  00000008     D
DS$V_CMDFLG                     =  00000007     D
DS$V_CTRLC                      =  00000000     D
DS$V_CTRLO                      =  00000010     D
DS$V_DEVFLG                     =  00000009     D
DS$V_DISABLCC                   =  00000018     D
DS$V_DONFLG                     =  0000000D     D
DS$V_ERRFLG                     =  00000004     D
DS$V_EXCEPT                     =  00000013     D
DS$V_EXETST                     =  00000012     D
DS$V_HLTFLG                     =  00000003     D
DS$V_LODFLG                     =  00000001     D
DS$V_MEMMGT                     =  0000000F     D
DS$V_OUTPUT                     =  00000017     D
DS$V_RUBFLG                     =  00000005     D
DS$V_SCRIPT                     =  00000015     D
DS$V_SETIMR                     =  00000019     D
DS$V_STRFLG                     =  00000002     D
DS$V_SUBT                       =  0000000E     D
DS$V_SYSFLG                     =  0000000A     D
DS$V_TIMRON                     =  00000011     D
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSA$M_LOAD_DEBUGGER             =  40000000     D
DSA$V_APT                       =  0000001F     D
DSA$V_DEBUG                     =  0000001A     D
DSA$V_LOAD_DEBUGGER             =  0000001E     D
DSA$V_QA                        =  0000000F     D
DSA$V_USER                      =  0000001C     D
DSP$GEN_PTABLES                    ********  X   03
DSQA$K_DISPAT_AFTER_INIT        =  00000014     D
DSQA$K_DISPAT_BEFORE_INIT       =  00000013     D
DSQA$K_DISPAT_CALLTEST          =  00000015     D
DSQA$K_DISPAT_DONE_TESTS        =  00000018     D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =  00000001     D
DSQA$K_DISPAT_DSX$ENDPASS_END   =  00000002     D
DSQA$K_DISPAT_DSX$ENDPASS_MID   =  00000000     D
DSQA$K_DISPAT_DS_CLEANUP_BGN    =  00000003     D
DSQA$K_DISPAT_DS_CLEANUP_END    =  00000004     D
DSQA$K_DUMMY_T                  =  00000007     D
DSQA$K_ERROR_ERROR_BGN          =  00000006     D
DSQA$K_ERROR_ERROR_END          =  00000005     D
DSQA$K_KERNEL_INIT_CONTEXT      =  00000008     D
DSQA$K_LOOP_DSX$BGNSUB          =  00000009     D
DSQA$K_LOOP_DSX$CKLOOP          =  0000000B     D
DSQA$K_LOOP_DSX$ENDSUB          =  0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS     =  0000000E     D
DSQA$K_PARAM_DSX$GETDATA        =  0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL     =  0000000F     D
DSQA$K_PARAM_DSX$GETSTRING      =  00000010     D
DSQA$K_PARAM_DSX$GETVIELD       =  0000000D     D
DSQA$K_QA_DSX$BRANCH            =  00000011     D
DSQA$K_START_BEFORE_HEADER      =  00000017     D
DSQA$K_START_VRRESTART          =  00000012     D
DSQA$K_START_VRSTART            =  00000016     D
DSR$FIND_USER_PC                   ********  X   03
DSV$DEBUG                          ********  X   03
DSX$ABORT                          000003A9 RG  D  03
DSX$PRINT                          ********  X   03
DS_CLEANUP                         ********  X   03
DS_TESTID                          ********  X   03
ENV$M_DOMAIN                    =  00000002     D
ENV$M_LEVEL                     =  00000001     D
ENV$M_SUPER                     =  000003FC     D
ENV$S_DOMAIN                    =  00000001     D
ENV$S_LEVEL                     =  00000001     D
ENV$S_SUPER                     =  00000008     D
ENV$V_DOMAIN                    =  00000001     D
ENV$V_LEVEL                     =  00000000     D
ENV$V_SUPER                     =  00000002     D
ENV$_CPU                        =  00000000     D
ENV$_FUNCTIONAL                 =  00000000     D
ENV$_REPAIR                     =  00000001     D
ENV$_SUPER                      =  00000001     D
ENV$_SYSTEM                     =  00000001     D
FALSE                           =  00000000     D
FMT_ABORT                          00000046 R   D  02
FMT_ABORT_NO_PC                    00000077 R   D  02
FMT_AMBSEC                         0000014C R   D  02
FMT_ILLSEC                         0000011E R   D  02
FMT_MISMATCH                       000001B6 R   D  02
FMT_NODEVICE                       000001F4 R   D  02
FMT_NOUNITS                        000000CB R   D  02
FMT_PROG                           00000006 R   D  02
FMT_SECTIONS                       0000017C R   D  02
FMT_START                          00000104 R   D  02
FMT_TESTING                        000001A3 R   D  02
FMT_TESTRANGE                      000000A0 R   D  02
INIT_CONTEXT                       ********      X  03
IS_IT_USER_PC                      000003D3 R   D  03
L$A_CCP                            00000240     D
L$A_DEVP                           0000021C     D
L$A_DREG                           00000224     D
L$A_DTP                            00000218     D
L$A_ICP                            0000023C     D
L$A_LASTADR                        00000214     D
L$A_NAME                           00000208     D
```

```
L$A_REPP                        00000244    D
L$A_SECNAM                      00000250    D
L$A_STATAB                      00000248    D
L$A_TSTCNT                      00000254    D
L$L_ENVIRON                     00000204    D
L$L_ERRTYP                      0000024C    D
L$L_HEADLENGTH                  00000200    D
L$L_REV                         0000020C    D
L$L_UNIT                        00000220    D
L$L_UNUSED                      00000228    D
L$L_UPDATE                      00000210    D
MAPFREE                         ********    X   03
MAP_DEBUGGER                    ********    X   03
OFF                           = 00000000    D
ON                            = 00000001    D
PRLIST                          00000362 R  D   03
QA$AOB_CHECK_STATE              ********    X   03
QA$AOB_FLAGS                    ********    X   03
QA$K_ABORT_NOW                = 00000007    D
QA$K_APT_PHASE_ONE            = 0000000B    D
QA$K_APT_PHASE_TWO            = 0000000C    D
QA$K_BAD_ADDRESS              = 00000009    D
QA$K_CONTROL_C_CONT           = 0000000A    D
QA$K_DEALLOCATION             = 0000000E    D
QA$K_ERROR_PHASE_ONE          = 00000005    D
QA$K_ERROR_PHASE_THREE        = 00000007    D
QA$K_ERROR_PHASE_TWO          = 00000006    D
QA$K_FAILURE                  = 00000000    D
QA$K_FIRST_BRANCH_CODE        = 00000000    D
QA$K_FIRST_TIME               = 00000005    D
QA$K_INIT_CONTEXT_DONE        = 00000003    D
QA$K_LAST_BRANCH_CODE         = 00000025    D
QA$K_LOOP_ON_SUBTEST          = 00000003    D
QA$K_LOOP_ON_TEST             = 00000002    D
QA$K_LOOP_TEST_DONE           = 00000004    D
QA$K_MEMORY_MANAGE            = 0000000D    D
QA$K_MULTIPLE_PASS            = 00000001    D
QA$K_NODEF                    = 00000000    D
QA$K_NORMAL_START             = 00000000    D
QA$K_NO_HEADER                = 00000006    D
QA$K_NUMBER_OF_CHECKS         = 0000000F    D
QA$K_NUMBER_QA_FLAGS          = 00000008    D
QA$K_NUMB_ROUTINE_STATES      = 00000004    D
QA$K_QA_DEBUG                 = 00000001    D
QA$K_QA_DONE                  = 00000002    D
QA$K_RUN_BACKWARDS            = 00000004    D
QA$K_SECTION                  = 00000008    D
QA$K_SUCCESS                  = 00000001    D
QA$MAIN                         ********    X   03
QA$V_CHECK_DONE               = 00000003    D
QA$V_DOING_CLEANUP            = 00000002    D
QA$V_ERROR_FOUND              = 00000001    D
QA$V_INIT_DONE                = 00000000    D
QIO$ADDUNIT                     ********    X   03
QIO$CLEANUP                     ********    X   03
SCB$L_ACCESS                    00000020    D
SCB$L_ARITH                     00000034    D
SCB$L_BREAK                     0000002C    D
SCB$L_CHME                      00000044    D
SCB$L_CHMK                      00000040    D
SCB$L_CHMS                      00000048    D
SCB$L_CHMU                      0000004C    D
SCB$L_COMPAT                    00000030    D
SCB$L_KNLSTK                    00000008    D
SCB$L_MACHCHK                   00000004    D
SCB$L_OPCCUS                    00000014    D
SCB$L_OPCDEC                    00000010    D
SCB$L_POWER                     0000000C    D
SCB$L_RADRMOD                   0000001C    D
SCB$L_ROPRAND                   00000018    D
SCB$L_RXDB                      000000F8    D
SCB$L_SFTLVL1                   00000084    D
SCB$L_SFTLVL10                  000000A8    D
SCB$L_SFTLVL11                  000000AC    D
SCB$L_SFTLVL12                  000000B0    D
SCB$L_SFTLVL13                  000000B4    D
SCB$L_SFTLVL14                  000000B8    D
SCB$L_SFTLVL15                  000000BC    D
SCB$L_SFTLVL2                   00000088    D
SCB$L_SFTLVL3                   0000008C    D
SCB$L_SFTLVL4                   00000090    D
SCB$L_SFTLVL5                   00000094    D
SCB$L_SFTLVL6                   00000098    D
SCB$L_SFTLVL7                   0000009C    D
SCB$L_SFTLVL8                   000000A0    D
SCB$L_SFTLVL9                   000000A4    D
SCB$L_TBIT                      00000028    D
SCB$L_TIMER                     000000C0    D
SCB$L_TRANSL                    00000024    D
SCB$L_TXDB                      000000FC    D
SCB$L_ZERO                      00000000    D
SCRIPT$FLUSH                    ********    X   03
SEP_FUNCTIONAL                = 00000002    D
SEP_REPAIR                    = 00000003    D
SIZ...                        = 00000001    D
START1                          00000051 R  D   03
TRUE                          = 00000001    D
VRABORT                         000003CF RG D   03
VRABORT_COM                     000003E3 R  D   03
VRRESTART                       00000051 RG D   03
VRSHOWSECTIONS                  000004DC RG D   03
VRSTART                         00000000 RG D   03
VRSTART_X                       0000033F R  D   03
VRSUPPORT                       00000482 RG D   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	0000021E (542.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	0000052E (1326.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000002	956 (4)	#-328 (3) 487 (3) 818 (4) 956 (4) #-356 (3) 517 (3) #-418 (3) #-537 (3) #-877 (4) #-892 (4) #-432 (3) 800 (4) #-945 (4)
$ER	=00000001	224 (3)	
$MODULE	00000000-R	224 (3)	
ACT_DEFAULT_DBG	00000000-XR		334 (3)
APC$_ABORT	=00000006	203 (3)	
APC$_CONTINUE	=00000009	203 (3)	
APC$_DUMP	=00000003	203 (3)	
APC$_NOP	=00000000	203 (3)	
APC$_START	=00000005	203 (3)	
APC$_ZERO	=00000004	203 (3)	
APM$_ABTDON	=00000006	203 (3)	#-838 (4)
APM$_DEVERR	=00000002	203 (3)	
APM$_DONE	=0000000A	203 (3)	
APM$_EXCEPT	=0000000B	203 (3)	
APM$_HARDERR	=00000003	203 (3)	
APM$_MORE	=00000008	203 (3)	
APM$_NOMES	=00000000	203 (3)	
APM$_PREPERR	=0000000C	203 (3)	
APM$_PRGERR	=00000005	203 (3)	
APM$_SOFTERR	=00000004	203 (3)	
APM$_SPOOL	=00000009	203 (3)	
APM$_SYSERR	=00000001	203 (3)	
BEGIN	00000000-XR		643 (3) 840 (4)
BIT...	=00000004	208 (3)	197 (3) 199 (3) 205 (3) 206 (3) 207 (3) 208 (3)
CEP_FUNCTIONAL	=00000000	199 (3)	
CEP_REPAIR	=00000001	199 (3)	
CHECK_QA_BIT	00000345-R	654 (3)	#-338 (3) #-360 (3)
CLI$K_ALNUM	=00000087	197 (3)	
CLI$K_ALPHA	=00000086	197 (3)	
CLI$K_BIF	=00000083	197 (3)	
CLI$K_BIFS	=00000094	197 (3)	
CLI$K_BR	=00000082	197 (3)	
CLI$K_BUFSIZ	=00000100	204 (3)	204 (3)
CLI$K_CALL	=00000092	197 (3)	
CLI$K_COMMA	=0000008E	197 (3)	
CLI$K_DEC	=0000008A	197 (3)	
CLI$K_EOL	=00000091	197 (3)	
CLI$K_ERROR	=00000080	197 (3)	
CLI$K_EXIT	=00000081	197 (3)	
CLI$K_FILE	=00000095	197 (3)	
CLI$K_HEX	=00000089	197 (3)	
CLI$K_KEYWORD	=0000008C	197 (3)	
CLI$K_NUM	=00000085	197 (3)	
CLI$K_OCT	=00000088	197 (3)	
CLI$K_RETURN	=00000093	197 (3)	
CLI$K_SIZE	00000444	204 (3)	

```
CLI$K_SLASH                     =00000090      197     (3)
CLI$K_SPACE                     =00000084      197     (3)
CLI$K_STRING                    =0000008B      197     (3)
CLI$K_SYMBOL                    =0000008D      197     (3)
CLI$K_VALUE                     =0000008F      197     (3)
CLI$L_ADDRESS                    00000018      204     (3)
CLI$L_COMMAND                    00000004      204     (3)
CLI$L_DATA                       0000001C      204     (3)
CLI$L_FLAGS                      00000000      204     (3)     407     (3)     412     (3)     429     (3)     657     (3)
CLI$L_LAST                       00000024      204     (3)   #-466     (3)
CLI$L_NEXT                       00000030      204     (3)
CLI$L_PASS                       0000002C      204     (3)   #-458     (3)
CLI$L_SUBT                       00000028      204     (3)   #-480     (3)
CLI$L_TEST                       00000020      204     (3)   #-460     (3)
CLI$Q_BUFQWD                     00000C34      204     (3)
CLI$Q_FILE                       00000008      204     (3)
CLI$Q_SECTION                    00000010      204     (3)   #-381     (3)   #-391     (3)   #-392     (3)
CLI$Q_TIME                       0000043C      204     (3)
CLI$T_BUFFER                     0000003C      204     (3)
CLI$V_ADAPTER                   =00000018      204     (3)
CLI$V_ADR                       =0000000B      204     (3)
CLI$V_ASCII                     =00000013      204     (3)
CLI$V_BREAK                     =0000000A      204     (3)
CLI$V_BRIEF                     =0000001B      204     (3)
CLI$V_BYTE                      =0000000D      204     (3)
CLI$V_CLEAR                     =00000002      204     (3)
CLI$V_DEC                       =00000010      204     (3)
CLI$V_DEFAULT                   =0000000C      204     (3)
CLI$V_DEPOSIT                   =00000019      204     (3)
CLI$V_EVENT                     =00000008      204     (3)
CLI$V_EXAM                      =00000005      204     (3)
CLI$V_FLAGS                     =00000009      204     (3)
CLI$V_HEX                       =00000012      204     (3)
CLI$V_KERNEL                    =00000017      204     (3)
CLI$V_LOAD                      =00000006      204     (3)
CLI$V_LONG                      =0000000F      204     (3)
CLI$V_NOTNUF                    =00000001      204     (3)
CLI$V_OCT                       =00000011      204     (3)
CLI$V_PREG                      =0000001A      204     (3)
CLI$V_QA                        =00000007      204     (3)   #-656     (3)
CLI$V_QACKLOOPLOOPS             =0000001C      204     (3)
CLI$V_QAERRORPRINTS             =0000001B      204     (3)
CLI$V_QAMULTIPLEPASS            =0000001F      204     (3)
CLI$V_QASUBTESTLOOPS            =0000001E      204     (3)
CLI$V_QATESTLOOPS               =0000001D      204     (3)
CLI$V_REG                       =00000014      204     (3)
CLI$V_REQUIRED                  =00000000      204     (3)
CLI$V_RUN                       =00000015      204     (3)
CLI$V_SET                       =00000003      204     (3)
CLI$V_SHOW                      =00000004      204     (3)
CLI$V_VALSEC                    =00000016      204     (3)   #-406     (3)   #-411     (3)   #-428     (3)
CLI$V_WORD                      =0000000E      204     (3)
DISPATCH                         00000000-XR                    635     (3)
DS$CLRVEC                        00000000-XR                    587     (3)
DS$GB_TYPECODE                   00000000-XR                  #-421     (3)   #-435     (3)   #-540     (3)   #-896     (4)
                                                              #-898     (4)   #-959     (4)
DS$GL_CLIBASE                    00000000-XR                    379     (3)     655     (3)
```

```
ZZ-ENSAA-7.0    Cross reference                                  27-JUL-1984 15:45:52  VAX-11 Macro V03-01          Page  41
START                          *** START Handle START command   23-MAY-1984 14:15:55  DMA1:[SYSO.SYSMAINT]START.MAR;220 (4)
Cross reference

DS$GL_FLAGS                     00000000-XR                      325     (3)    #-449    (3)    #-625    (3)    #-748    (4)
                                                                #-783    (4)      785    (4)      830    (4)    #-836    (4)
                                                                  874    (4)      942    (4)
DS$GL_FSTTEST                   00000000-XR                     #-461    (3)    #-464    (3)    #-473    (3)
DS$GL_LSTTEST                   00000000-XR                     #-467    (3)    #-471    (3)    #-476    (3)
DS$GL_NUMTEST                   00000000-XR                     #-456    (3)    #-470    (3)    #-474    (3)    #-477    (3)
                                                                #-487    (3)
DS$GL_SUBTEST                   00000000-XR                     #-481    (3)
DS$GPQARD                       00000000-XR                      564     (3)      594    (3)
DS$GQ_TESTID                    00000000-XR                     #-800    (4)    #-818    (4)
DS$K_BBC                        =0000001D       207     (3)
DS$K_BBCC                       =00000021       207     (3)
DS$K_BBCCI                      =00000023       207     (3)
DS$K_BBCS                       =0000001F       207     (3)
DS$K_BBS                        =00000C1C       207     (3)
DS$K_BBSC                       =00000020       207     (3)
DS$K_BBSS                       =0000001E       207     (3)
DS$K_BBSSI                      =00000022       207     (3)
DS$K_BCC_M                      =0000001B       207     (3)
DS$K_BCS_M                      =0000001A       207     (3)
DS$K_BEQLU_B                    =00000005       207     (3)
DS$K_BEQLU_M                    =00000011       207     (3)
DS$K_BEQL_B                     =00000004       207     (3)
DS$K_BEQL_M                     =00000010       207     (3)
DS$K_BERROR                     =00000026       207     (3)
DS$K_BGEQU_B                    =00000007       207     (3)
DS$K_BGEQU_M                    =00000013       207     (3)
DS$K_BGEQ_B                     =00000006       207     (3)
DS$K_BGEQ_M                     =00000012       207     (3)
DS$K_BGTRU_B                    =00000009       207     (3)
DS$K_BGTRU_M                    =00000015       207     (3)
DS$K_BGTR_B                     =00000008       207     (3)
DS$K_BGTR_M                     =00000014       207     (3)
DS$K_BLBC                       =00000025       207     (3)      207    (3)
DS$K_BLBS                       =00000024       207     (3)
DS$K_BLEQU_B                    =00000003       207     (3)
DS$K_BLEQU_M                    =0000000F       207     (3)
DS$K_BLEQ_B                     =00000002       207     (3)
DS$K_BLEQ_M                     =0000000E       207     (3)
DS$K_BLSSU_B                    =00000001       207     (3)
DS$K_BLSSU_M                    =0000000D       207     (3)
DS$K_BLSS_B                     =00000000       207     (3)      207    (3)
DS$K_BLSS_M                     =0000000C       207     (3)
DS$K_BNEQU_B                    =0000000B       207     (3)
DS$K_BNEQU_M                    =00000017       207     (3)
DS$K_BNEQ_B                     =0000000A       207     (3)
DS$K_BNEQ_M                     =00000016       207     (3)
DS$K_BNERROR                    =00000027       207     (3)
DS$K_BVC_M                      =00000019       207     (3)
DS$K_BVS_M                      =00000018       207     (3)
DS$K_DS_STARTING_LOCATION       =00010000       207     (3)
DS$K_PRINIB                     =00000002       208     (3)
DS$K_PRINTF                     =00000001       208     (3)  #-328    (3)    #-356    (3)    #-418    (3)    #-432    (3)
                                                             #-487    (3)    #-517    (3)    #-537    (3)    #-570    (3)
                                                             #-693    (3)    #-800    (4)    #-818    (4)    #-877    (4)
                                                             #-892    (4)    #-945    (4)    #-956    (4)
DS$K_PRINTI                     =00000000       208     (3)
```

```
DS$K_PRINTX                     =00000003       208     (7)
DS$K_TYPE_ABORT_PROGRAM         =00000014       208     (3)     #-800   (4)     #-818   (4)
DS$K_TYPE_ABORT_TEST            =00000013       208     (3)
DS$K_TYPE_COMMAND_ERR           =00000015       208     (3)     #-328   (3)     #-418   (3)     #-432   (3)     #-487   (3)
                                                                #-877   (4)     #-945   (4)
DS$K_TYPE_COMMAND_OUT           =00000016       208     (3)     #-892   (4)     #-895   (4)     #-956   (4)
DS$K_TYPE_CRD_AUTOTEST          =0000001A       208     (3)
DS$K_TYPE_DS_PROMPT             =00000001       208     (3)
DS$K_TYPE_DS_START              =0000001D       208     (3)
DS$K_TYPE_ERRDEV                =00000008       208     (3)
DS$K_TYPE_ERRHARD               =00000006       208     (3)
DS$K_TYPE_ERROR_BODY            =00000009       208     (3)
DS$K_TYPE_ERROR_END             =0000000A       208     (3)
DS$K_TYPE_ERRPREP               =0000001B       208     (3)
DS$K_TYPE_ERRSOFT               =00000007       208     (3)
DS$K_TYPE_ERRSUP                =00000004       208     (3)
DS$K_TYPE_ERRSYS                =00000005       208     (3)
DS$K_TYPE_ERR_HALT              =0000000D       208     (3)
DS$K_TYPE_EXCEPTION             =0000000C       208     (3)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B       208     (3)
DS$K_TYPE_FIRST_PASS            =00000011       208     (3)
DS$K_TYPE_GENERAL               =00000000       208     (3)
DS$K_TYPE_GENERAL_ERROR         =00000003       208     (3)
DS$K_TYPE_NO_TESTS              =00000012       208     (3)
DS$K_TYPE_PARAM_ERROR           =0000001C       208     (3)
DS$K_TYPE_PROGRAM_END           =00000010       208     (3)
DS$K_TYPE_PROGRAM_INFO          =00000017       208     (3)     #-571   (3)
DS$K_TYPE_PROGRAM_START         =0000000F       208     (3)     #-517   (3)
DS$K_TYPE_QIO_INVADP            =00000024       208     (3)
DS$K_TYPE_QIO_NODRIVER          =00000022       208     (3)
DS$K_TYPE_QIO_WRONGVER          =00000023       208     (3)
DS$K_TYPE_SCRIPT_ECHO           =00000021       208     (3)
DS$K_TYPE_SCRIPT_PNF            =0000001E       208     (3)
DS$K_TYPE_SCRIPT_PROMPT         =00000020       208     (3)
DS$K_TYPE_SCRIPT_SKIP           =0000001F       208     (3)
DS$K_TYPE_SEQUENCE_ERROR        =00000019       208     (3)
DS$K_TYPE_START_ERR             =00000018       208     (3)     #-356   (3)     #-537   (3)
DS$K_TYPE_START_LIST            =00000025       208     (3)     #-694   (3)
DS$K_TYPE_SUMMARY               =0000000E       208     (3)
DS$K_TYPE_USER_PROMPT           =00000002       208     (3)
DS$M_ABRTFLG                    =00000040       205     (3)
DS$M_BADTIME                    =00100000       205     (3)
DS$M_BATCH                      =00400000       205     (3)
DS$M_BRKCLR                     =00001000       205     (3)
DS$M_BRKPT                      =00000800       205     (3)     #-447   (3)     #-746   (4)     #-781   (4)
DS$M_CHARFLG                    =00000100       205     (3)
DS$M_CMDFLG                     =00000080       205     (3)
DS$M_CTRLC                      =00000001       205     (3)     #-446   (3)
DS$M_CTRLO                      =00010000       205     (3)
DS$M_DEVFLG                     =00000200       205     (3)
DS$M_DISABLCC                   =01000000       205     (3)
DS$M_DONFLG                     =00002000       205     (3)     #-443   (3)
DS$M_ERRFLG                     =00000010       205     (3)
DS$M_EXCEPT                     =00080000       205     (3)     #-448   (3)     #-747   (4)     #-782   (4)
DS$M_EXETST                     =00040000       205     (3)
DS$M_HLTFLG                     =00000008       205     (3)     #-445   (3)     #-745   (4)     #-780   (4)
DS$M_LODFLG                     =00000002       205     (3)
```

```
DS$M_MEMMGT                     =00008000      205     (3)
DS$M_OUTPUT                     =00800000      205     (3)
DS$M_RUBFLG                     =00000020      205     (3)
DS$M_SCRIPT                     =00200000      205     (3)
DS$M_SETIMR                     =02000000      205     (3)
DS$M_STRFLG                     =00000004      205     (3)    #-444    (3)    #-624    (3)    #-835    (4)
DS$M_SUBT                       =00004000      205     (3)
DS$M_SYSFLG                     =00000400      205     (3)
DS$M_TIMRON                     =00020000      205     (3)
DS$V_ABRTFLG                    =00000006      205     (3)
DS$V_BADTIME                    =00000014      205     (3)
DS$V_BATCH                      =00000016      205     (3)
DS$V_BRKCLR                     =0000000C      205     (3)
DS$V_BRKPT                      =0000000B      205     (3)
DS$V_CHARFLG                    =00000C08      205     (3)
DS$V_CMDFLG                     =00000007      205     (3)
DS$V_CTRLC                      =00000000      205     (3)
DS$V_CTRLO                      =00000010      205     (3)
DS$V_DEVFLG                     =00000009      205     (3)
DS$V_DISABLCC                   =00000018      205     (3)
DS$V_DONFLG                     =0000000D      205     (3)
DS$V_ERRFLG                     =00000004      205     (3)    #-829    (4)
DS$V_EXCEPT                     =00000013      205     (3)
DS$V_EXETST                     =00000012      205     (3)
DS$V_HLTFLG                     =00000003      205     (3)
DS$V_LODFLG                     =00000001      205     (3)    #-324    (3)    #-873    (4)    #-941    (4)
DS$V_MEMMGT                     =0000000F      205     (3)
DS$V_OUTPUT                     =00000017      205     (3)
DS$V_RUBFLG                     =00000005      205     (3)
DS$V_SCRIPT                     =00000015      205     (3)
DS$V_SETIMR                     =00000019      205     (3)
DS$V_STRFLG                     =00000002      205     (3)    #-784    (4)
DS$V_SUBT                       =0000000E      205     (3)
DS$V_SYSFLG                     =0000000A      205     (3)
DS$V_TIMRON                     =00000011      205     (3)
DSA$GL_FLAGS                     0000FE00                      331    (3)     333    (3)    #-337    (3)     372    (3)
                                                               497    (3)     551    (3)     581    (3)     584    (3)
                                                               659    (3)     664    (3)     741    (4)     828    (4)
DSA$GL_MSGTYP                    0000FE40                    #-839    (4)
DSA$GL_PASSES                    0000FE08                    #-459    (3)
DSA$GL_PASSNO                    0000FE54                    #-800    (4)    #-818    (4)
DSA$GL_SECTNO                    0000FE10                    #-380    (3)    #-403    (3)
DSA$GL_UNITS                     0000FE0C                    #-525    (3)    #-532    (3)    #-567    (3)    #-600    (3)
DSA$M_LOAD_DEBUGGER             =40000000                    #-336    (3)
DSA$V_APT                       =0000001F                    #-331    (3)    #-827    (4)
DSA$V_DEBUG                     =0000001A                    #-583    (3)
DSA$V_LOAD_DEBUGGER             =0000001E                    #-332    (3)
DSA$V_QA                        =0000000F                    #-496    (3)    #-550    (3)    #-658    (3)    #-663    (3)
                                                             #-740    (4)
DSA$V_USER                      =0000001C                    #-371    (3)    #-580    (3)
DSP$GEN_PTABLES                  00000000-XR                   530    (3)
DSQA$K_DISPAT_AFTER_INIT        =00000014      206     (3)
DSQA$K_DISPAT_BEFORE_INIT       =00000013      206     (3)
DSQA$K_DISPAT_CALLTEST          =00000015      206     (3)
DSQA$K_DISPAT_DONE_TESTS        =00000018      206     (3)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001      206     (3)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002      206     (3)
```

```
DSQA$K_DISPAT_DSX$ENDPASS_MID  =00000000     206     (3)
DSQA$K_DISPAT_DS_CLEANUP_BGN   =00000003     206     (3)
DSQA$K_DISPAT_DS_CLEANUP_END   =00000004     206     (3)
DSQA$K_DUMMY_T                 =00000007     206     (3)
DSQA$K_ERROR_ERROR_BGN         =00000006     206     (3)
DSQA$K_ERROR_ERROR_END         =00000005     206     (3)
DSQA$K_KERNEL_INIT_CONTEXT     =00000008     206     (3)
DSQA$K_LOOP_DSX$BGNSUB         =00000009     206     (3)
DSQA$K_LOOP_DSX$CKLOOP         =0000000B     206     (3)
DSQA$K_LOOP_DSX$ENDSUB         =0000000A     206     (3)
DSQA$K_PARAM_DSX$GETADDRESS    =0000000E     206     (3)
DSQA$K_PARAM_DSX$GETDATA       =0000000C     206     (3)
DSQA$K_PARAM_DSX$GETLOGICAL    =0000000F     206     (3)
DSQA$K_PARAM_DSX$GETSTRING     =00000010     206     (3)
DSQA$K_PARAM_DSX$GETVIELD      =00000COD     206     (3)
DSQA$K_QA_DSX$BRANCH           =00000011     206     (3)
DSQA$K_START_BFFORE_HEADER     =00000017     206     (3)    #-494    (3)
DSQA$K_START_VRRESTART         =00000012     206     (3)    #-363    (3)
DSQA$K_START_VRSTART           =00000016     206     (3)    #-339    (3)
DSR$FIND_USER_PC                00000000-XR                    771    (4)
DSV$DEBUG                       00000000-XR                    335    (3)
DSX$ABORT                       000003A9-R   738     (4)
DSX$PRINT                       00000000-XR                    328    (3)     356    (3)     418    (3)     432    (3)
                                                               487    (3)     517    (3)     537    (3)     572    (3)
                                                               695    (3)     800    (4)     818    (4)     877    (4)
                                                               892    (4)     945    (4)     956    (4)
DS_CLEANUP                      00000000-XR                    369    (3)     833    (4)
DS_TESTID                       00000000-XR                    787    (4)
ENV$M_DOMAIN                   =00000002     199     (3)
ENV$M_LEVEL                    =00000001     199     (3)
ENV$M_SUPER                    =000003FC     199     (3)
ENV$S_DOMAIN                   =00000001     199     (3)
ENV$S_LEVEL                    =00000001     199     (3)
ENV$S_SUPER                    =00000008     199     (3)    #-350    (3)     #-881    (4)     #-949    (4)
ENV$V_DOMAIN                   =00000001     199     (3)      199    (3)
ENV$V_LEVEL                    =00000000     199     (3)      199    (3)
ENV$V_SUPER                    =00000002     199     (3)    #-349    (3)     #-880    (4)     #-948    (4)
ENV$_CPU                       =00000000     199     (3)      199    (3)
ENV$_FUNCTIONAL                =00000000     199     (3)      199    (3)
ENV$_REPAIR                    =00000001     199     (3)      199    (3)
ENV$_SUPER                     =00000001     199     (3)    #-352    (3)     #-883    (4)     #-951    (4)
ENV$_SYSTEM                    =00000001     199     (3)      199    (3)
FALSE                          =00000000     207     (3)
FMT_ABORT                       00000046-R   230     (3)      800    (4)
FMT_ABORT_NO_PC                 00000077-R   233     (3)      818    (4)
FMT_AMBSEC                      0000014C-R   254     (3)      418    (3)
FMT_ILLSEC                      0000011E-R   251     (3)      432    (3)
FMT_MISMATCH                    000001B6-R   263     (3)      356    (3)
FMT_NODEVICE                    000001F4-R   266     (3)      892    (4)
FMT_NOUNITS                     000000CB-R   245     (3)      537    (3)
FMT_PROG                        00000006-R   226     (3)      517    (3)
FMT_SECTIONS                    0000017C-R   257     (3)      956    (4)
FMT_START                       00000104-R   248     (3)      328    (3)     877    (4)     945    (4)
FMT_TESTING                     000001A3-R   260     (3)      569    (3)
FMT_TESTRANGE                   000000A0-R   242     (3)      487    (3)
INIT_CONTEXT                    00000000-XR                    359    (3)     623    (3)     832    (4)
IS_IT_USER_PC                   000003D3-R   768     (4)    #-752    (4)
```

```
L$A_DEVP                        0000021C                        #-538   (3)    #-885   (4)
L$A_NAME                        00000208                        #-517   (3)    #-956   (4)
L$A_SECNAM                      00000250                        #-385   (3)    #-402   (3)    #-419   (3)    #-433   (3)
                                                                #-957   (4)
L$A_TSTCNT                      00000254                        #-455   (3)    #-517   (3)
L$L_ENVIRON                     00000204                          351   (3)      589   (3)      882   (4)      950   (4)
L$L_REV                         0000020C                        #-517   (3)
L$L_UNIT                        00000220                        #-526   (3)
L$L_UPDATE                      00000210                        #-517   (3)
MAPFREE                         00000000-XR                       582   (3)
MAP_DEBUGGER                    00000000-XR                       585   (3)
OFF                             =00000000       207     (3)
ON                              =00000001       207     (3)
PRLIST                          00000362-R      677     (3)     #-420   (3)    #-434   (3)    #-539   (3)    #-897   (4)
                                                                #-958   (4)
QA$AOB_CHECK_STATE              00000000-XR                       499   (3)      502   (3)      553   (3)      556   (3)
                                                                  743   (4)
QA$AOB_FLAGS                    00000000-XR                       507   (3)      561   (3)
QA$K_ABORT_NOW                  =00000007       207     (3)
QA$K_APT_PHASE_ONE              =0000000B       207     (3)
QA$K_APT_PHASE_TWO              =0000000C       207     (3)
QA$K_BAD_ADDRESS                =00000009       207     (3)
QA$K_CONTROL_C_CONT             =0000000A       207     (3)
QA$K_DEALLOCATION               =0000000E       207     (3)
QA$K_ERROR_PHASE_ONE            =00000005       207     (3)     #-501   (3)    #-555   (3)    #-742   (4)
QA$K_ERROR_PHASE_THREE          =00000007       207     (3)
QA$K_ERROR_PHASE_TWO            =00000006       207     (3)
QA$K_FAILURE                    =00000000       207     (3)
QA$K_FIRST_BRANCH_CODE          =00000000       207     (3)
QA$K_FIRST_TIME                 =00000005       207     (3)
QA$K_INIT_CONTEXT_DONE          =00000003       207     (3)
QA$K_LAST_BRANCH_CODE           =00000025       207     (3)
QA$K_LOOP_ON_SUBTEST            =00000003       207     (3)     #-498   (3)    #-552   (3)
QA$K_LOOP_ON_TEST               =00000002       207     (3)
QA$K_LOOP_TEST_DONE             =00000004       207     (3)
QA$K_MEMORY_MANAGE              =0000000D       207     (3)
QA$K_MULTIPLE_PASS              =00000001       207     (3)
QA$K_NODEF                      =00000000       207     (3)
QA$K_NORMAL_START               =00000000       207     (3)
QA$K_NO_HEADER                  =00000006       207     (3)     #-506   (3)    #-560   (3)
QA$K_NUMBER_OF_CHECKS           =0000000F       207     (3)
QA$K_NUMBER_QA_FLAGS            =00000008       207     (3)
QA$K_NUMB_ROUTINE_STATES        =00000004       207     (3)
QA$K_QA_DEBUG                   =00000001       207     (3)
QA$K_QA_DONE                    =00000002       207     (3)
QA$K_RUN_BACKWARDS              =00000004       207     (3)
QA$K_SECTION                    =00000008       207     (3)
QA$K_SUCCESS                    =00000001       207     (3)
QA$MAIN                         00000000-XR                       339   (3)      363   (3)      494   (3)
QA$V_CHECK_DONE                 =00000003       207     (3)
QA$V_DOING_CLEANUP              =00000002       207     (3)
QA$V_ERROR_FOUND                =00000001       207     (3)
QA$V_INIT_DONE                  =00000000       207     (3)
QIO$ADDUNIT                     00000000-XR                       598   (3)
QIO$CLEANUP                     00000000-XR                       373   (3)
SCB$L_TIMER                     000000C0                        #-587   (3)
SCRIPT$FLUSH                    00000000-XR                       348   (3)      779   (4)
```

SEP_FUNCTIONAL	=00000002	199	(3)	#-590	(3)						
SEP_REPAIR	=00000003	199	(3)								
SIZ...	=00000001	205	(3)	199	(3)	205	(3)				
START1	00000051-R	347	(3)								
TRUE	=00000001	207	(3)								
VRABORT	000003CF-R	759	(4)								
VRABORT_COM	000003E3-R	773	(4)	#-770	(4)						
VRRESTART	00000051-R	346	(3)	#-749	(4)						
VRSHOWSECTIONS	000004DC-R	939	(4)								
VRSTART	00000000-R	323	(3)								
VRSTART_X	0000033F-R	642	(3)	#-329	(3)	#-357	(3)	#-422	(3)	#-436	(3)
				#-488	(3)	#-541	(3)	#-599	(3)		
VRSUPPORT	00000482-R	872	(4)								

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_PRINT_S	1	328 (3)	328 (3) 356 (3) 418 (3) 432 (3) 487 (3) 517 (3) 537 (3) 800 (4) 818 (4) 877 (4) 892 (4) 945 (4) 956 (4)
$DEF	1	208 (3)	
$DEFINI	1	196 (3)	196 (3) 198 (3) 200 (3) 201 (3) 202 (3)
$DS_CFDEF	1	196 (3)	196 (3)
$DS_CLIDEF	1	197 (3)	197 (3)
$DS_CLRVEC_S	1	587 (3)	587 (3)
$DS_DSADEF	5	198 (3)	198 (3)
$DS_ENVDEF	2	199 (3)	199 (3)
$DS_GPHARD_S	1	564 (3)	564 (3) 594 (3)
$DS_HDRDEF	2	200 (3)	200 (3)
$DS_QADEF	2	207 (3)	207 (3)
$DS_SCBDEF	3	201 (3)	201 (3)
$DS_TYPEDEF	4	208 (3)	208 (3)
$EQU	1	208 (3)	197 (3) 199 (3) 206 (3) 207 (3) 208 (3)
$EQULS1	1	208 (3)	197 (3) 199 (3) 206 (3) 207 (3) 208 (3)
$EQULST	1	197 (3)	197 (3) 199 (3) 206 (3) 207 (3) 208 (3)
$GBLINI	2		197 (3) 199 (3) 205 (3) 206 (3) 207 (3) 208 (3)
$PRDEF	4	202 (3)	202 (3)
$PRINT	2	326 (3)	326 (3) 354 (3) 416 (3) 430 (3) 484 (3) 510 (3) 535 (3) 795 (4) 814 (4) 875 (4) 890 (4) 943 (4) 953 (4)
$VIELD	1		199 (3) 205 (3)
$VIELD1	1	208 (3)	199 (3) 205 (3)
APTDEF	1	203 (3)	203 (3)
BR_IF_APT	1	331 (3)	331 (3)
CLIDEF	3	204 (3)	204 (3)
DSFDEF	3	205 (3)	205 (3)
DSQA	2	206 (3)	206 (3)
DS_QADEFS	6	207 (3)	207 (3)
MODNAM	1	224 (3)	224 (3)
QA_MAIN	1	339 (3)	339 (3) 363 (3) 494 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.12	00:00:00.30
Command processing	143	00:00:00.83	00:00:01.74
Pass 1	950	00:00:15.25	00:00:24.53
Symbol table sort	0	00:00:00.80	00:00:00.88
Pass 2	267	00:00:03.59	00:00:04.67
Symbol table output	57	00:00:00.34	00:00:00.64
Psect synopsis output	7	00:00:00.04	00:00:00.07
Cross-reference output	124	00:00:01.77	00:00:02.15
Assembler run totals	1585	00:00:22.75	00:00:34.99

The working set limit was 1000 pages.
102702 bytes (201 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 573 non-local and 47 local symbols.
1007 source lines were read in Pass 1, producing 0 object records in Pass 2.
114 pages of virtual memory were used to define 28 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DRR1:[DS.WORK]DIAG.MLB;955	11
DRB1:[DS.WORK]DS.MLB;218	9
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	26

732 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) START/UPDA=(START.UPD,START.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMAINT
```

Table of contents

(2) 37 DECLARATIONS
(3) 303.1 TS11/TS04, TU80 Bootstrap driver

```
                        0000     .1          .TITLE  TSBTDRIVR      *** TSBTDRIVR driver for TS11/TS04, TU80 magtape
                        0000     .2          .IDENT  /V01-02/
-2                      0000      3          .DISABLE GLOBAL
                        0000      4
                        0000      5 ;
                        0000     .1 ;                          COPYRIGHT (c) 1980,1982 BY
-1                      0000      7 ;                DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        0000      8 ;
                        0000      9 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
                        0000     10 ; ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
                        0000     11 ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
                        0000     12 ; COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
                        0000     13 ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
                        0000     14 ; TRANSFERRED.
                        C000     15 ;
                        0000     16 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
                        C000     17 ; AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT B.' DIGITAL EQUIPMENT
                        0000     18 ; CORPORATION.
                        0C00     19 ;
                        0000     20 ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
                        0000     21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        0000     22 ;
                        0000     23
                        0000     24 ;++
                        0000     25 ; FACILITY:     DIAGNOSTIC SUPERVISOR
                        0000     26 ;
                        0000     27 ; ABSTRACT:
                        0000     .1 ;       This module contains the file load driver for
                        0000     .2 ;       UNIBUS TS11/TS04 and TU80 magtape
-2                      0000     30 ;
                        0000     31 ; ENVIRONMENT:  kernel mode
                        0000     32 ;
                        0000     33 ; AUTHOR:  Roger Riggs, CREATION DATE:  2-OCT-1980
                        0000     34 ;
                        0000     .1 ; MODIFIED BY:
                        0000     .2 ;
                        0000     .3 ;       01      Bob Bergazzi    Aug 30, 1982    v6.9
                        0000     .4 ;               Changed MS$CMD to reside all on one page. Routine
                        0000     .5 ;               MAP_VA won't work if MS$CMD is split over a page boundary
                        0000     .6 ;
                        0000     .7 ;       02      Marion Baggett  Dec 01, 1982    V6.10
                        0000     .8 ;               Added comment changes for TU80.
                        0000     .9
                        0000     35 ;--
```

```
                        0000     37          .SBTTL  DECLARATIONS
                        0000     38 ;
                        0000     39 ; INCLUDE FILES:
                        0000     40 ;
                        0000     41
                        0000     42          $BTDDEF                         ; Define boot device types
                        0000     43          $IODEF                          ; Define I/O function codes
                        0000     44          $NDTDEF                         ; Define NEXUS device types
                        0000     45          $PTEDEF                         ; Define page table entry
                        0000     46          $PRDEF                          ; Define processor registers
                        0000     47          $RPBDEF                         ; Define RPB offsets
                        0000     48          $SSDEF                          ; Define system status codes
                        0000     49          $VADEF                          ; Define virtural address fields
                        0000     50
                        0000     51 ;
                        0000     52 ; MACROS:
                        0000     53 ;
                        0000     54
                        0000     55
                        0000     56 ;
                        0000     57 ; EQUATED SYMBOLS.
                        0000     58 ;
                000000B1 0000    59 BTD$K_TM = 177 ; Temp
                000000B2 0000    60 btd$k_TS = 178 ; Temp
                        0000     61
                        0000     62 ;
                        0000     63 ; LOCAL SYMBOLS
                        0000     64 ;
                        0000     65
                        0000     66 ;
                        0000     .1 ; TS11/TS04, TU80 COMMAND PACKET DEFINITION
-1                      0000     68 ;
                        0000     69 ;
                        0000     70
                        0000     71          $DEFINI MS
                        0000     72
                        0000     73
                00000000 0000    74 .=0                                      ;RESET PC??????
                        0000     75 $DEF     MS_CPHD .BLKW   1               ;COMMAND PACKET HEADER
                        0002     76          _VIELD  MS_CPHD,0,<-            ;
                        0002     77                  <COD,5>,-               ;COMMAND CODE FIELD
                        0002     78                  <,2>,-                  ;B5-B6 ALWAYS 0 FOR TS04
                        0002     79                  <IE,,M>,-               ;INTERRUPT ENABLE
                        0002     80                  <MOD,4>,-               ;COMMAND MODE FIELD(B11-B8)
                        0002     81                                          ; B8=REVERSE & B9=RETRY
                        0002     82                  <SWB,,M>,-              ;SWAP BYTES BIT(B12)
                        0002     83                  <OPP,,M>,-              ;OPPOSITE BIT(B13)
                        0002     84                  <CVC,,M>,-              ;CLEAR VOLUME CHECK(B14)
                        0002     85                  <ACK,,M>,-              ;ACKNOWLEDGE BIT(B15)
                        0002     86          >
                        0002     87 $DEF     MS_BACT .BLKW   1               ;BUS ADDRESS(B15-B0) OR COUNT
                        0004     88 $DEF     MS_BA1  .BLKW   1               ;BUS ADDRESS B17-B16(RIGHT JUST)
                        0006     89 $DEF     MS_CNT  .BLKW   1               ;BYTE COUNT
                        0008     90                                          ;FOR WRITE CHARACTERISTIC DATA
                        0008     91 $DEF     MS_MBA0 .BLKW   1               ;MESSAGE BUFFER ADDR. WRD 1
                        000A     92 $DEF     MS_MBA1 .BLKW   1               ;MESSAGE BUFFER ADDR. WRD 2
                        000C     93 $DEF     MS_LNTH .BLKW   1               ;MESSAGE BUFFER LENGTH(ALWAYS 14.)
```

```
                     94 $DEF     MS_CHWD .BLKW   1               ;CHARACTERISTIC WORD
                     95          _VIELD  MS_CHWD,4,<-            ;
                     96                  <ERI,,M>,-              ;ENABLE MESSAGE BUFFER RELEASE INTERRUPTS
                     97                  <EAI,,M>,-              ;ENABLE ATTENTION INTERRUPTS
                     98                  <ENB,,M>,-              ;USED WITH ESS BIT***
                     99                  <ESS,,M>,-              ;ENABLE SKIP TAPE MARKS STOP
                    100                 >
                    101
                    102 ;
                      .1 ; TS11/TS04,TU80  MESSAGE PACKET DEFINITION
-1                  104 ;
                    105
                    106
                    107 $DEF     MS_MHD  .BLKW   1               ;MESSAGE PACKET
                    108          _VIELD  MS_MHD,0,<-             ;MESSAGE HEADER WORD
                    109                  <COD,5>,-               ;MESSAGE CODE FIELD
                    110                  <FMT,3>,-               ;FORMAT FIELD
                    111                  <CLS,4>,-               ;CLASS CODE FIELD
                    112                  <RSR,3>,-               ;RESERVED FIELD
                    113                  <ACK,,M>,-              ;MESSAGE ACKNOWLEDGE BIT(B15)
                    114                 >                        ;
                    115 $DEF     MS_LNH  .BLKW   1               ;MESSAGE LENGTH WORD
                    116                                          ;HIGH BYTE=0,LOW BYTE=1010(LENGTH)
                    117 $DEF     MS_RBPC .BLKW   1               ;RESIDUAL BYTE/POSITION COUNT
                    118 $DEF     MS_XSR0 .BLKW   1               ;EXTENDED STATUS REGISTER 0
                    119          _VIELD  MS_XSR0,0,<-            ;
                    120                  <EOT,,M>,-              ;END OF TAPE DETECTED(B0)
                    121                  <BOT,,M>,-              ;BEGINNING OF TAPE(B1)
                    122                  <WLK,,M>,-              ;WRITE LOCKED(B2)
                    123                  <PED,,M>,-              ;PHASE ENCODED DRIVE(B3)
                    124                  <VCK,,M>,-              ;VOLUME CHECK(B4)
                    125                  <IE,,M>,-               ;INTERRUPT WAS ENABLED(B5)
                    126                  <ONL,,M>,-              ;DEVICE ON-LINE(B6)
                    127                  <MOT,,M>,-              ;TAPE MOVING ON LAST COMMAND(B7)
                    128                  <ILA,,M>,-              ;ILLEGAL ADDRESS(B8)
                    129                  <ILC,,M>,-              ;ILLEGAL COMMAND(B9)
                    130                  <NEF,,M>,-              ;NON-EXECUTALBE FUNCTION(B10)
                    131                  <WLE,,M>,-              ;WRITE LOCK ERROR(B11)
                    132                  <RLL,,M>,-              ;RECORD LENGTH LONG(B12)
                    133                  <LET,,M>,-              ;LOGICAL END OF TAPE(B13)
                    134                  <RLS,,M>,-              ;RECORD LENGTH SHORT(B14)
                    135                  <TMK,,M>,-              ;TAPE MARK DETECTED(B15)
                    136                 >                        ;
                    137 $DEF     MS_XSR1 .BLKW   1               ;EXTENDED STATUS REGISTER 1
                    138          _VIELD  MS_XSR1,0,<-            ;
                    139                  <MTE,,M>,-              ;(PE) MULTI-TRACK ERROR
                    140                                          ;(NRZ) VERTICAL PARITY ERROR
                    141                  <UNC,,M>,-              ;(PE) UNCORRECTABLE DATA ERROR(B1)
                    142                                          ;(NRZ) CYCLIC REDUNDANCY CHECK ERROR
                    143                  <POL,,M>,-              ;(PE) POSTAMBLE LONG(B2)
                    144                                          ;(NRZ) LONGITUDINAL REDUNDANCY CHECK ERROR
                    145                  <POS,,M>,-              ;(PE) POSTAMBLE SHORT(B3)
                    146                                          ;(NRZ) NOISE RECORD
                    147                  <IED,,M>,-              ;(PE) INVALID END DATA(B4)
                    148                                          ;(NRZ) LRC WAS 0.
                    149                  <IPO,,M>,-              ;(PE) INVALID POSTAMBLE(B5)
                    150                                          ;(NRZ) ILLEGAL TAPE MARK
```

```
                     001A   151                     <SYN,,M>,-              ;(PE) SYNCH ERROR(B6)
                     001A   152                             -               ;(NRZ) FRAME DROPOUT
                     001A   153                     <IPR,,M>,-              ;(PE) INVALID PREAMBLE(B7)
                     001A   154                     <,1>,-                  ;RESERVED BIT
                     001A   155                     <SCK,,M>,-              ;SPEED CHECK(B9)
                     001A   156                     <DBF,,M>,-              ;(PE) DESKEW BUFFER FAIL(B10)
                     001A   157                             -               ;(NRZ) NRZ BOARD FIFO OVERFLOW
                     001A   158                     <TIG,,M>,-              ;TRASH IN GAP(B11)
                     001A   159                     <CRS,,M>,-              ;CREASE DETECTED(B12)
                     001A   160                     <COR,,M>,-              ;CORRCTABLE DATA(B13)
                     001A   161                     <,1>,-                  ;UNUSED BIT(B14)
                     001A   162                     <DLT,,M>,-              ;DATA LATE(B15)
                     001A   163             >
                     001A   164 $DEF         MS_XSR2 .BLKW   1               ;EXTENDED STATUS REGISTER 2
                     001C   165             _VIELD  MS_XSR2,0,<-            ;
                     001C   166                     <DTP,8>,-               ;DEAD TRACK PARITY,B7-B0
                     001C   167                     <XSK,,M>,-              ;EXCESSIVE SKEW(B9)
                     001C   168                     <WCF,,M>,-              ;WRITE CLOCK FAIL(B10),BROKEN HARDWARE
                     001C   169                     <,1>,-                  ;B11 NOT USED
                     001C   170                     <CAF,,M>,-              ;CAPSTAN ACCELERATION FAIL(B12)
                     001C   171                     <BPE,,M>,-              ;SERIAL BUS PARITY ERROR AT DRIVE(B13)
                     001C   172                     <SIP,,M>,-              ;SILO PARITY ERROR(B14)
                     001C   173                     <OPM,,M>,-              ;OPERATION IN PROGRESS(B15)
                     001C   174             >
                     001C   175 $DEF         MS_XSR3 .BLKW   1               ;EXTENDED STATUS REGISER 3
                     001E   176             _VIELD  MS_XSR3,0,<-            ;
                     001E   177                     <RIB,,M>,-              ;REVERSE INTO BOT(B0)
                     001E   178                     <LXS,,M>,-              ;LIMIT EXCEEDED STATICALLY(B1)
                     001E   179                     <NOI,,M>,-              ;NOISE RECORD(B2)
                     001E   180                     <DCK,,M>,-              ;DENSITY CHECK(B3)
                     001E   181                     <CRF,,M>,-              ;CAPSTAN RESPONSE FAIL(B4)
                     001E   182                     <REV,,M>,-              ;TAPE MOVED BACKWARDS(B5)
                     001E   183                     <OPI,,M>,-              ;OPERATION IN COMPLETE(B6)
                     001E   184                     <LMX,,M>,-              ;TAPE LIMIT EXCEEDED(B7)
                     001E   185                     <FEC,8>,-               ;B15-B8, FATAL ERROR CODE(U-DIAGNOSTIC)
                     001E   186             >
                     001E   187
                     001E   188             $DEFEND MS
                     0000   189
                     0000   190 ;
                     0000     1 ; TS11/TS04,TU80 TSSR TERMINATION CLASS CODES
-1                   0000   192 ;
                     0000   193
            00000000 0000   194 TCC_NML=0                                   ;NORAML TERMINATION
            00000001 0000   195 TCC_ATN=1                                   ;ATTENTION CONDITION
            00000002 0000   196 TCC_TSA=2                                   ;TAPE STATUS ALERT
            00000003 0000   197 TCC_FNR=3                                   ;FUNCTION REJECT
            00000004 0000   198 TCC_REM=4                                   ;RECOVERABLE ERROR(TAPE MOVED)
            00000005 0000   199 TCC_REN=5                                   ;RECOVERABLE ERROR(TAPE NOT MOVED)
            00000006 0000   200 TCC_UER=6                                   ;UNRECOVERABLE ERROR(TAPE POSI. LOST)
            00000007 0000   201 TCC_FTL=7                                   ;FATAL CONTROLLER ERROR
                     0000   202
                     0000   203 ;
                     0000   204 ; FATAL CLASS (FC) CODES IN TSSR
                     0000   205 ;
                     0000   206
            00000000 0000   207 FCC_IDF=0                                   ;INTERNAL DIAG. FAILURE
```

```
                    00000001  0000   208 FCC_CPE=1                                   ;IO SEQUENCE CROM PARITY ERROR
                    00000002  0000   209 FCC_UPE=2                                   ;U-PROCESSOR CROM PARITY ERROR OR OTHER
                    00000003  0000   210 FCC_LAP=3                                   ;LOSS OF AC POWER DETECTED
                              0000   211
                              0000   212 ;
                              0000    .1 ; TS11/TS04, TU80 MESSAGE CODES IN MS_MHD_COD
-1                            0000   214 ;
                              0000   215
                    00000010  0000   216 MSG_END=^0020                               ;END
                    00000011  0000   217 MSG_FAL=^0021                              ;FAIL
                    00000012  0000   218 MSG_ERR=^0022                              ;ERROR
                    00000013  0000   219 MSG_ATN=^0023                              ;ATTENTION
                    00000014  0000   220 MSG_LOG=^0024                              ;LOG (NOT USED)
                              0000   221
                              C000   222 ;
                              0000   223 ; CLASS CODE FOR MESSAGE CODES (MS_MHD_CLS VALUES)
                              0000   224 ;
                              0000   225
                              0000   226                                  ;**WHEN MSG TYPE=ATTENTION**
                    00000000  0000   227 CLS_ONF=0                                  ;ON OR OFFLINE
                    00000001  0000   228 CLS_MDF=1                                  ;MICRO DIAG. FAILURE
                              0000   229                                  ;**WHEN MSG TYPE=FAIL**
                    00000000  0000   230 CLS_PTB=0                                  ;PACKET BAD(SERIAL BUS PARITY ERROR)
                    00000001  0000   231 CLS_OTHER=1                                ;OTHERS
                    00000002  0000   232 CLS_WLN=2                                  ;WRITE LOCK ERROR OR NON-EXECUTABLE FUNCTION
                    00000003  0000   233 CLS_MDE=3                                  ;MICRO DIAGNOSTIC ERROR
                              0000   234
                              0000   235 ;
                              0000    .1 ; TS11/TS04, TU80 HARDWARE COMMAND MODES/CODES
-1                            0000   237 ;
                              0000   238
                              0000   239                                      ; INTERRUPT ENABLE & ACKNOWLEDGE
                    00000000  0000   240 HC_NOP=0                          ;SIMULATED NOP(REAL NO OPERATION)
                    00000000  0000   241 HC_PAK=HC_NOP                              ;SIMULATED PACK ACKNOWLEDGE
                    00000000  0000   242 HC_WCK=HC_NOP                              ;SIMULATED WRITE CHECK
                    00000000  0000   243 HC_WKR=HC_NOP                              ;SIMULATED WRITE CHECK REVERSE
                    00000000  0000   244 HC_RPS=HC_NOP                              ;SIMULATED READ IN PRESET
                    00000000  0000   245 HC_SCH=HC_NOP                              ;SIMULATED SET CHARACTERISTICS
                              0000   246
                              0000   247
                    0000C001  0000   248 HC_RDN=^00001!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;* READ NEXT (FORWARD)
                    0000C004  0000   249 HC_WRC=^00004!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;* WRITE CHARACTERISTICS
                    0000C008  0000   250 HC_SRF=^00010!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;$ SPACE RECORDS FORWARD
                    0000C108  0000   251 HC_SRR=^00410!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;$ SPACE RECORDS REVERSE
                    0000C208  0000   252 HC_STF=^01010!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;$ SKIP TAPE MARKS FORWARD
                    0000C308  0000   253 HC_STR=^01410!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;$ SKIP TAPE MARKS REVERSE
                    0000C408  0000   254 HC_RWD=^02010!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;$ REWIND
                    0000C10A  0000   255 HC_UNL=^00412!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;- REWIND AND UNLOAD
                    0000C00B  0000   256 HC_DRI=^00013!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;- DRIVER INITIALIZE
                    0000C00F  0000   257 HC_GST=^00017!MS_CPHD_M_CVC!MS_CPHD_M_ACK        ;- GET STATUS IMMEDIATE
                              0000   258                                  ;**NOTE**
                              0000   259                                  ; * => DATA XFR
                              0000   260                                  ; + => (SPECIAL)
                              0000   261                                  ; $ => POSITION
                              0000   262                                  ; - => FORMAT,CONTROL,INITIALIZE,& STATUS
                              0000   263 ;*
                              0000   264 ; TSSR defs
```

```
                                0000   265 ;-
                                0000   266
                    00000000    0000   267 TSDB    = 0                     ; Offset to TSDB
                    00000000    0000   268 TSBA    = 0                     ; Offset to TSBA
                    00000002    0000   269 TSSR    = 2                     ; Offset to TSSR
                                0000   270
                                0000   271     $FIELD  MS_TSSR,1,<-        ;TS11/TS04, TU80 STATUS REGISTER(B0 UNUSED)
-1                              0000   272             <TCC,3>,-           ;TERMINATION CLASS CODE FIELD
                                0000   273             <FC,2>,-            ;FATAL ERROR CLASS CODE FIELD
                                0000   274             <OFL,,M>,-          ;DEVICE IS OFF-LINE(B6)
                                0000   275             <SSR,,M>,-          ;SUBSYSTEM READY(B7)
                                0000   276             <A16,,M>,-          ;BUFFER ADDRESS BIT 16
                                0000   277             <A17,,M>,-          ;BUFFER ADDRESS BIT 17
                                0000   278             <NBA,,M>,-          ;NEED BUFFER ADDRESS(B10)
                                0000   279             <NXM,,M>,-          ;NON-EXISTENT MEMORY(B11)
                                0000   280             <RMR,,M>,-          ;REGISTER MODIFICATION REFUSED(B12)
                                0000   281             <SPE,,M>,-          ;SERIAL BUS PARITY ERROR(B13)
                                0000   282             <UPE,,M>,-          ;UNIBUS PARITY ERROR(B14)
                                0000   283             <SC,,M>,-           ;SPECIAL CONDITION(B15)
                                0000   284             >                   ;
                                0000   285
                                0000   286 ;
                                0000   287 ; OWN
                                0000   288 ;
                            00000000    1           .PSECT  DATA, SHR, NOEXE, NOWRT, 5
-2                  00000020    0000   291 MS$CMD::         .BLKW   16      ; Command packet/char data/message area
                                0020   292
                                0020   293
                                0020   294 ;
                                0020   295 ; BOOT DRIVER TABLE ENTRY
                                0020   296 ;
                                0020   297
                                0020     1         $BOOT_DRIVER    DEVTYPE = BTD$K_TS,-    ; Device type (UNIBUS)
-1                              0020   299                         SIZE = TS11_DRVSIZ,-    ; Driver size
                                0020   300                         ADDR = TS11_DRIVER,-    ; Driver address
                                0020   301                         ENTRY = TS1T_DRIVER,-   ; Entry point
                                0020   302                         DRIVRNAME = DRIVERNAME  ; Driver name
```

```
                                .SBTTL  TS11/TS04, TU80 Bootstrap driver
-1
                        ;++
                        ; Functional description:
                        ;
                        ;       This routine performs basic magnetic tape positioning.
                        ;       functions are supported for read, rewind, skip files (forward
                        ;       backward) and skip records (forward and backward). It also
                        ;       support drive clear.  Drive clear must be called before the first
                        ;       tape operation as it is necessary to initialize the tape subsystem
                        ;       prior to the first position or transfer operation.
                        ;       Drive clear initialization clears the Need Buffer Address (NBA)
                        ;       bit in the drive. This bit can be reset by a UNIBUS reset/init
                        ;       which in the TS11 causes the tape to be rewound. If NBA is set
                        ;       the function is rejected with tape position lost.
                        ;
                        ; Inputs:
                        ;
                        ;       R3      - base address of adapter's register space
                        ;       R5      - LBN for current piece of transfer
                        ;       R6      - contains 0
                        ;       R7      - Address of devices CSR
                        ;       R8      - size of transfer in bytes
                        ;       R9      - address of the RPB
                        ;       R10     - starting address of transfer (byte offset in first
                        ;                 page ORed with starting map register number)
                        ;
                        ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_REWIND only)
                        ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                        ;       SIZE(AP)- Size of buffer in bytes
                        ;       BUF(AP) - Address of buffer
                        ;
                        ; Implicit inputs:
                        ;
                        ;       RPB$B_SLAVE     - Slave number of drive
                        ;       RPB$W_UNIT      - RPB field containing boot device unit number
                        ;
                        ;
                        ; Outputs:
                        ;
                        ;       R0 - status code
                        ;
                        ;               SS$_NORMAL      - successful transfer
                        ;               SS$_TAPEPOSLOST - Tape position lost
                        ;               SS$_TIMEOUT     - Device timeout
                        ;               SS$_DATAOVERUN  - Record too long
                        ;               SS$_ILLIOFUNC   - Illegal I/O function
                        ;               SS$_CTRLERR     - Controller error
                        ;               SS$_ENDOFFILE   - Tape mark was read
                        ;               SS$_DEVOFFLINE  - Device is offline
-1
                        ;
                        ; Register usage:
                        ;
                        ;               R5      - Map number of map for command area
                        ;               R11     - Address of command buffer
                        ;
                        ;--
```

```
                    0000   355
00000004            0000   356 BUF = 4
00000008            0000   357 SIZE = 8
0000000C            0000   358 LBN = 12
00000010            0000   359 FUNC = 16
00000014            0000   360 MODE = 20
```

```
                                0000    .1 TS11_DRIVER::                        ; TS11/TS04, TU80  driver
-1                              0000   363
                    04 AE   D4  0000   364          CLRL    4(SP)               ; Clear retry count ****************
                 52   02 A7   B0  0003   365          MOVW    TSSR(R7), R2        ; Get current status
                 0C 52    0A   E1  0007   366          BBC     #MS_TSSR_V_NBA, R2, 10$ ; Branch if not reset since last op
                 04   10 AC   D1  000B   367          CMPL    FUNC(AP),#IO$_DRVCLR ; Drive clear?
                         06   13  000F   368          BEQL    10$                 ; If so continue
               50   0224 8F   3C  0011   369          MOVZWL  #SS$_TAPEPOSLOST, R0 ; Assume tape position lost
                              05  0016   370          RSB
                              0017   371
                    09   00   EF  0017   372 10$:     EXTZV   #VA$V_BYTE, #VA$S_BYTE,-
                    55   5A        001A   373                  R10, R5            ; Byte offset from start
                55   03FF C845   9E  001C   374          MOVAB   ^X3FF(R8)[R5],R5    ; Length of transfer + extra page
                  55   F7 8F   78  0022   375          ASHL    #-9, R5, R5         ; Number of pages to mapped + 1
                         55         0026
                              0027   376
                00000000'8F   D0  0027   377          MOVL    #MS$CMD, R11        ; Address of command area
                         5B        002D
                    15   09   EF  002E   378          EXTZV   #VA$V_VPN, #VA$S_VPN,-
                    50   5B        0031   379                  R11, R0            ; Get PFN to map
                51   0800 C345   DE  0033   380          MOVAL   ^X800(R3)[R5], R1   ; Address of map register
                 90000000 E0    9E  0039   381          MOVAB   <PTE$M_VALID!PTE$C_KW>(R0), -
                         61        003F
                              0040   382                  (R1)                ; Map first page
              81   81   01   C1  0040   383          ADDL3   #1, (R1)+, (R1)+    ; Map 2nd page
                         61   D4  0044   384          CLRL    (R1)                ; And clear 3rd
                              0046   385
                    8000 8F   B0  0046   386          MOVW    #MS_MHD_M_ACK,MS_MHD(R11); Set ACK in message buffer
                      10 AB       004A
                50    10 AB   3E  004C   387          MOVAW   MS_MHD(R11), R0     ; Copy message buffer address
                     013C     30  0050   388          BSBW    MAP_VA              ; Get UNIBUS address of this VA
                              0053   389
              08 AB    50   D0  0053   390          MOVL    R0, MS_MBA0(R11)    ; Set address of message buffer
              CC AB    0E   B0  0057   391          MOVW    #14, MS_LNTH(R11)   ; Length of message buffer
                 0E AB        B4  005B   392          CLRW    MS_CHWD(R11)        ; Characteristics word
                              005E   393
            6B   C004 8F   B0  005E   394          MOVW    #HC_WRC, MS_CPHD(R11) ; Write characteristics command
              50   08 AB   9E  0063   395          MOVAB   MS_MBA0(R11), R0    ; Address of characteristic data
                     0125     30  0067   396          BSBW    MAP_VA              ; Get UNIBUS address of this VA
              02 AB    50   D0  006A   397          MOVL    R0, MS_BACT(R11)    ; Set low and high address
              06 AB    08   B0  006E   398          MOVW    #8, MS_CNT(R11)     ; Set length of area
                              0072   399
                     010C     30  0072   400          BSBW    EXECUTE             ; Execute function
                     00A5     30  0075   401          BSBW    WAIT                ; Wait for Subsystem Ready to set
                    25 50     E9  0078   402          BLBC    R0,RETURN           ; Exit if subsystem ready does not set
                              0078   403
                              0078   404
                              0078   405 ;+
                              0078   406 ; branch on function type
                              0078   407 ;-
            21    10 AC   D1  0078   408          CMPL    FUNC(AP),#IO$_READLBLK  ; Read logical block?
                     2F   13  007F   409          BEQL    E_READ              ; Execute read function
            24    10 AC   D1  0081   410          CMPL    FUNC(AP),#IO$_REWIND    ; Rewind?
                     1A   13  0085   411          BEQL    E_REWIND            ; Do a rewind
            25    10 AC   D1  0087   412          CMPL    FUNC(AP),#IO$_SKIPFILE  ; Skip past eof
                     4C   13  008B   413          BEQL    E_SKIPFILE          ; Do it
            26    10 AC   D1  008D   414          CMPL    FUNC(AP),#IO$_SKIPRECORD; Skip records?
```

```
                68   13  0091   415          BEQL    E_SKIPRECORD                    ; Do it
             50 01   D0  0093   416          MOVL    #SS$_NORMAL, R0                 ; Assume normal completion
          04   10 AC   D1  0096   417          CMPL    FUNC(AP),#IO$_DRVCLR            ; Drive clear?
                04   13  009A   418          BEQL    RETURN                          ; If so have already done it
          50   F4 8F   9A  009C   419          MOVZBL  #SS$_ILLIOFUNC,R0               ; illegal I/O function
                       00A0   420 RETURN:
                05     00A0   421          RSB                                     ; Return to caller
                       00A1   422 ;+
                       00A1   423 ; Rewind slave drive, wait for completion
                       00A1   424 ;-
                       00A1   425 E_REWIND:
       6B   C408 8F    B0  00A1   426          MOVW    #HC_RWD, MS_CPHD(R11)           ; Set function to rewind
             50   5B   D0  00A6   427          MOVL    R11, R0                         ; VA to map
                00D5    30  00A9   428          BSBW    EXECUTE                         ; Execute function
                006E    30  00AC   429          BSBW    WAIT                            ; Wait for ready to sec
                05     00AF   430          RSB                                     ; Exit
                       00B0   431 ;+
                       00B0   432 ; Read block of data
                       00B0   433 ;-
                       00B0   434 E_READ:
       6B   C001 8F    B0  00B0   435          MOVW    #HC_RDN, MS_CPHD(R11)           ; Set function to read next
         02 AB   5A    D0  00B5   436          MOVL    R10, MS_BACT(R11)               ; Set address to read to
         06 AB   58    B0  00B9   437          MOVW    R8, MS_CNT(R11)                 ; Set length to read
                00C1    30  00BD   438          BSBW    EXECUTE                         ; Execute function
                005A    30  00C0   439          BSBW    WAIT                            ; Wait for completion
          51   14 AB   3C  00C3   440          MOVZWL  MS_RBPC(R11), R1                ; Bytes residual byte count
       51   58   51    C3  00C7   441          SUBL3   R1, R8, R1                      ; From desired length is transfered
             0A 50    E9  00CB   442          BLBC    R0, 20$                         ; Branch if complete failure
    05 16 AB    0C    E1  00CE   443          BBC     #MS_XSR0_V_RLL, MS_XSR0(R11), -
                       00D3   444                  20$                             ; Branch if record not longer than buf
        50   0838 8F    3C  00D3   445          MOVZWL  #SS$_DATAOVERUN, R0             ; Longer, change return
                05     00D8   446 20$:    RSB
                       00D9   447
                       00D9   448 ;+
                       00D9   449 ; Skip to end of file
                       00D9   450 ; P1 is negative if backwards else forwards
                       00D9   451 ;-
                       00D9   452 E_SKIPFILE:
             50   01    D0  00D9   453          MOVL    #SS$_NORMAL,R0                  ; Assume it works
       6B   C208 8F    B0  00DC   454          MOVW    #HC_STF, MS_CPHD(R11)           ; Set function to space tape marks
    02 AB   04 AC    F7  00E1   455          CVTLW   BUF(AP), MS_BACT(R11)           ; Set count of marks to skip
                0C    14  00E6   456          BGTR    10$                             ; Branch if positive
                10    13  00E8   457          BEQL    20$                             ; Branch if count 0
       6B   C308 8F    B0  00EA   458          MOVW    #HC_STR, MS_CPHD(R11)           ; Set function to space marks reverse
    02 AB   02 AB    AE  00EF   459          MNEGW   MS_BACT(R11),MS_BACT(R11); Make count positive
                       00F4   460
                008A    30  00F4   461 10$:    BSBW    EXECUTE                         ; Start function
                       00F7   462
                0023    30  00F7   463          BSBW    WAIT                            ; Wait for it to finish
                05     00FA   464 20$:    RSB
                       00FB   465
                       00FB   466 ;+
                       00FB   467 ; Skip to end of record, read w/o buffer
                       00FB   468 ; P1 is negative if backwards else forwards
                       00FB   469 ;-
                       00FB   470 E_SKIPRECORD:
             50   01    D0  00FB   471          MOVL    #SS$_NORMAL,R0                  ; Assume it works
```

```
                 6B   C008 8F    B0  00FE    472          MOVW    #HC_SRF, MS_CPHD(R11)   ; Assume forward
               02 AB    04 AC    F7  0103    473          CVTLW   BUF(AP), MS_BACT(R11)   ; Set record count
                           0C    14  0108    474          BGTR    30$                     ; All set, execute
                           10    13  010A    475          BFQL    50$                     ; Finish if space = 0
                 6B   C108 8F    B0  010C    476          MOVW    #HC_SRR, MS_CPHD(R11)   ; Function = space reverse
               02 AB    02 AB    AE  0111    477          MNEGW   MS_BACT(R11),MS_BACT(R11); Set record count
                                 0116    478
                         0068    30  0116    479 30$:     BSBW    EXECUTE                 ; Translate R11 command and execute
                         0001    30  0119    480          BSBW    WAIT                    ; Wait for it to finish
                                 011C    481
                                 05  011C    482 50$:     RSB
                                 011D    483
                                 011D    484
                                 011D    485 ;+
                                 C11D    486 ; Wait for slave to quiesce
                                 011D    487 ;-
                                 011D    488 WAIT:
                  3B9ACA00 8F    D0  011D    489          MOVL    #1000000000, R0         ; time out counter
                           50        0123
                       0080 8F    B3  0124    490 10$:     BITW    #MS_TSSR_M_SSR, TSSR(R7); Subsystem ready?
                         02 A7        0128
                            09    12  012A    491          BNEQ    20$                     ; Exit if subsystem ready
                          F5 50    F5  012C    492          SOBGTR  R0, 10$                 ; Count and try again
                     50   022C 8F    3C  012F    493          MOVZWL  #SS$_TIMEOUT, R0        ; Set return code
                                 05  0134    494          RSB                             ; return
                                 0135    495
                     50   0054 8F    3C  0135    496 20$:     MOVZWL  #SS$_CTRLERR, R0        ; Assume controller error
                     51   7C00 8F    B3  013A    497          BITW    #MS_TSSR_M_UPE ! MS_TSSR_M_SPE ! -
                                         013F    498                  MS_TSSR_M_RMR ! MS_TSSR_M_NXM ! -
                                         013F    499                  MS_TSSR_M_NBA, R1       ; Any fatal errors?
                                    3F   12  013F    500          BNEQ    90$                     ; Exit if any
                                         0141    501
                       51     02 A7    32  0141    502          CVTWL   TSSR(R7), R1            ; Get TSSR
                                    36   18  0145    503          BGEQ    80$                     ; Branch if SC clear, normal
                                         0147    504
                     51   03     01    EF  0147    505          EXTZV   #MS_TSSR_V_TCC, #MS_TSSR_S_TCC, -
                                    50       014B    506                  R1, R0                  ; Get termination class
                                             014C    507          CASE    R0, LIMIT = #TCC_NML, DISPLIST = <80$, 80$, 50$, 60$>
                     50   0054 8F    3C  0158    508          MOVZWL  #SS$_CTRLERR, R0        ; Assume controller error
                                    21   11  015D    509          BRB     90$                     ; And exit
                                         015F    510
                                         015F    511 ;+
                                         015F    512 ; Tape status condition
                                         015F    513 ;-
                           8003 8F    B3  015F    514 50$:     BITW    #MS_XSR0_M_TMK ! -
                             16 AB        0163
                                         0165    515                  MS_XSR0_M_BOT ! -
                                         0165    516                  MS_XSR0_M_EOT, -
                                         0165    517                  MS_XSR0(R1)             ;Tape mark detected?
                                    16   13  0165    518          BEQL    80$                     ; No, exit normal
                     50   0870 8F    3C  0167    519          MOVZWL  #SS$_ENDOFFILE,R0       ; Assume EOF sensed
                                    12   11  016C    520          BRB     90$                     ; Exit
                                         016E    521
                                         016E    522 ;+
                                         016E    523 ; Function reject
                                         016E    524 ;-
                                         016E    525 60$:
```

```
                50   84 8F    9A  016E    526          MOVZBL  #SS$_DEVOFFLINE,R0      ; Assume offline
          09 16 AB    06    E1  0172    527          BBC     #MS_XSR0_V_ONL, -
                                0177    528                  MS_XSR0(R11), 90$       ; Branch if offline
                                0177    529
                50    54 8F    9A  0177    530          MOVZBL  #SS$_CTRLERR,R0         ; Assume error
                         03    11  017B    531          BRB     90$                     ; Exit
                                017D    532 ;+
                                017D    533 ; Success
                                017D    534 ;-
                     50   01    D0  017D    535 80$:     MOVL    #SS$_NORMAL, R0         ; Success
                          05    05  0180    536 90$:     RSB                             ; Return from WAIT
                                0181    537
                                0181    538
                                0181    539 ;+
                                C181    540 ; Execute command pointer @R11
                                0181    541 ;-
                                0181    542 EXECUTE:
                     50   5B    D0  0181    543          MOVL    R11, R0                 ; VA to map
                          09    10  0184    544          BSBB    MAP_VA                  ; Map to UNIBUS space
              51    50   10    9C  0186    545          ROTL    #16, R0, R1             ; Position bits 16 & 17
          67  51    50   A9    9A  018A    546          BISW3   R0, R1, TSBA(R7)        ; Execute the function
                          05    05  018E    547          RSB
                                018F    548
                                018F    549 ;+
                                018F    550 ; Map and translate VA to UNIBUS address
                                018F    551 ; R0 is VA to be mapped to UNIBUS space
                                018F    552 ;       R0      VA to be mapped
                                018F    553 ;       R3      Base of adapter space
                                018F    554 ;       R5      Map register number assigned for command/message area
                                018F    555 ;               may require two maps
                                018F    556 ;-
                                018F    557 MAP_VA:
     00000000'8F    51    D0  018F    558          MOVL    #MS$CMD, R1             ; Get command address
                                0195
           51   F7 8F    51    78  0196    559          ASHL    #-9, R1,R1              ; PFN only
                                019A
                                019B    560
          50   15    09    ED  019B    561          CMPZV   #VA$V_VPN, #VA$S_VPN, -
                          51        019F    562                  R0, R1                  ; Does VA map same page?
                          07    12  01A0    563          BNEQ    10$                     ; Branch if should be other page
          15   09    55    F0  01A2    564          INSV    R5, #VA$V_VPN, #VA$S_VPN, -
                          50        01A6
                                01A7    565                  R0                      ; Insert UNIBUS PFN
                          15    11  01A7    566          BRB     20$                     ; Exit
                                01A9    567
                          51    D6  01A9    568 10$:     INCL    R1                      ; Next map maps next PFN
          54   15    09    ED  01AB    569          CMPZV   #VA$V_VPN, #VA$S_VPN, -
                          51        01AF    570                  R4, R1                  ; Does it map this page?
                          0D    12  01B0    571          BNEQ    30$                     ; Abort if page not mapped by either
          15   09    55    F0  01B2    572          INSV    R5, #VA$V_VPN, #VA$S_VPN, -
                          50        01B6
                                01B7    573                  R0                      ; Insert UNIBUS PFN
           00000200 8F    50    C0  01B7    574          ADDL    #^X200, R0              ; Map to next page
                                01BD
                                01BE    575
                          05    05  01BE    576 20$:     RSB                             ; Return
                                01BF    577
```

```
                     50    2C    3C  01BF   578 30$:    MOVZWL  #SS$_ABORT, R0          ; Set return code
                           8E    D5  01C2   579          TSTL    (SP)+                  ; Return extra return address
                                 05  01C4   580          RSB                            ; And return from bootdriver
                                     01C5   581
                                     01C5   582
                                     01C5   583 DRIVERNAME:                             ; Boot device driver name
                   49 52 44 53 54 00' 01C5    .1          .ASCIC  /TSDRIVER.EXE/         ; TSDRIVER
                   58 45 2E 52 45 56  01CB
                                 45  01D1
                                 0C  01C5
-1                                   01D2   585
                           000001D2  01D2   586 TS11_DRVSIZ = .-TS11_DRIVER
                                     01D2   587
                                     01D2   588          .END
```

```
$TABLE                  = 00000000 R   D  03
BIT...                  = 00000010     D
BTD$K_TM                = 000000B1     D
BTD$K_TS                = 000000B2     D
BUF                     = 00000004     D
CLS_MDE                 = 00000003     D
CLS_MDF                 = 00000001     D
CLS_ONF                 = 00000000     D
CLS_OTHER               = 00000001     D
CLS_PTB                 = 00000000     D
CLS_WLN                 = 00000002     D
DRIVERNAME                000001C5 R   D  04
EXECUTE                   00000181 R   D  04
E_READ                    000000B0 R   D  04
E_REWIND                  000000A1 R   D  04
E_SKIPFILE                000000D9 R   D  04
E_SKIPRECORD              000000FB R   D  04
FCC_CPE                 = 00000001     D
FCC_IDF                 = 00000000     D
FCC_LAP                 = 00000003     D
FCC_UPE                 = 00000002     D
FUNC                    = 00000010     D
HC_DRI                  = 0000C00B     D
HC_GST                  = 0000C00F     D
HC_NOP                  = 00000000     D
HC_PAK                  = 00000000     D
HC_RDN                  = 0000C001     D
HC_RPS                  = 00000000     D
HC_RWD                  = 0000C408     D
HC_SCH                  = 00000000     D
HC_SRF                  = 0000C008     D
HC_SRR                  = 0000C108     D
HC_STF                  = 0000C208     D
HC_STR                  = 0000C308     D
HC_UNL                  = 0000C10A     D
HC_WCK                  = 00000000     D
HC_WKR                  = 00000000     D
HC_WRC                  = 0000C004     D
IO$_DRVCLR              = 00000004     D
IO$_READLBLK            = 00000021     D
IO$_REWIND              = 00000024     D
IO$_SKIPFILE            = 00000025     D
IO$_SKIPRECORD          = 00000026     D
LBN                     = 0000000C     D
MAP_VA                    0000018F R   D  04
MODE                    = 00000014     D
MS$CMD                    00000000 RG  D  02
MSG_ATN                 = 00000013     D
MSG_END                 = 00000010     D
MSG_ERR                 = 00000012     D
MSG_FAL                 = 00000011     D
MSG_LOG                 = 00000014     D
MS_BA1                    00000004     D
MS_BACT                   00000002     D
MS_CHWD                   0000000E     D
MS_CNT                    00000006     D
MS_CPHD                   00000000     D
MS_CPHD_M_ACK           = 00008000     D
MS_CPHD_M_CVC           = 00004000     D
MS_LNH                    00000012     D
MS_LNTH                   0000000C     D
MS_MBA0                   00000008     D
MS_MBA1                   0000000A     D
MS_MHD                    00000010     D
MS_MHD_M_ACK            = 00008000     D
MS_RBPC                   00000014     D
MS_TSSR_M_A16           = 00000100     D
MS_TSSR_M_A17           = 00000200     D
MS_TSSR_M_NBA           = 00000400     D
MS_TSSR_M_NXM           = 00000800     D
MS_TSSR_M_OFL           = 00000040     D
MS_TSSR_M_RMR           = 00001000     D
MS_TSSR_M_SC            = 00008000     D
MS_TSSR_M_SPE           = 00002000     D
MS_TSSR_M_SSR           = 00000080     D
MS_TSSR_M_UPE           = 00004000     D
MS_TSSR_S_FC            = 00000002     D
MS_TSSR_S_TCC           = 00000003     D
MS_TSSR_V_A16           = 00000008     D
MS_TSSR_V_A17           = 00000009     D
MS_TSSR_V_FC            = 00000004     D
MS_TSSR_V_NBA           = 0000000A     D
MS_TSSR_V_NXM           = 0000000B     D
MS_TSSR_V_OFL           = 00000006     D
MS_TSSR_V_RMR           = 0000000C     D
MS_TSSR_V_SC            = 0000000F     D
MS_TSSR_V_SPE           = 0000000D     D
MS_TSSR_V_SSR           = 00000007     D
MS_TSSR_V_TCC           = 00000001     D
MS_TSSR_V_UPE           = 0000000E     D
MS_XSR0                   00000016     D
MS_XSR0_M_BOT           = 00000002     D
MS_XSR0_M_EOT           = 00000001     D
MS_XSR0_M_TMK           = 00008000     D
MS_XSR0_V_ONL           = 00000006     D
MS_XSR0_V_RLL           = 0000000C     D
MS_XSR1                   00000018     D
MS_XSR2                   0000001A     D
MS_XSR3                   0000001C     D
PTE$C_KW                = 10000000     D
PTE$M_VALID             = 80000000     D
RETURN                    000000A0 R   D  04
SIZ...                  = 00000001     D
SIZE                    = 00000008     D
SS$_ABORT               = 0000002C     D
SS$_CTRLERR             = 00000054     D
SS$_DATAOVERUN          = 00000838     D
SS$_DEVOFFLINE          = 00000084     D
SS$_ENDOFFILE           = 00000870     D
SS$_ILLIOFUNC           = 000000F4     D
SS$_NORMAL              = 00000001     D
SS$_TAPEPOSLOST         = 00000224     D
SS$_TIMEOUT             = 0000022C     D
TCC_ATN                 = 00000001     D
```

```
TCC_FNR                         =  00000003     D
TCC_FTL                         =  00000007     D
TCC_NML                         =  00000000     D
TCC_REM                         =  00000004     D
TCC_REN                         =  00000005     D
TCC_TSA                         =  00000002     D
TCC_UER                         =  00000006     D
TS1T_DRIVER                        00000000 RG  D  04
TS11_DRVSIZ                     =  000001D2     D
TSBA                            =  00000000     D
TSDB                            =  00000000     D
TSSR                            =  00000002     D
VA$S_BYTE                       =  00000009     D
VA$S_VPN                        =  00000015     D
VA$V_BYTE                       =  0000C000     D
VA$V_VPN                        =  00000009     D
WAIT                               0000011D R   D  04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000001E (30.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000020 (32.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC 21
BOOTDRIVR_4	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	000001D2 (466.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	302 (2)	302 (2)
BIT...	=00000010	284 (2)	284 (2)
BTD$K_TM	=000000B1	59 (2)	
BTD$K_TS	=000000B2	60 (2)	302 (2)
BUF	=00000004	356 (3)	#-455 (4) #-473 (4)
CLS_MDE	=00000003	233 (2)	
CLS_MDF	=00000001	228 (2)	
CLS_ONF	=00000C00	227 (2)	
CLS_OTHER	=00000001	231 (2)	
CLS_PTB	=00000000	230 (2)	
CLS_WLN	=00000002	232 (2)	
DRIVERNAME	000001C5-R	583 (4)	302 (2)
EXECUTE	00000181-R	542 (4)	#-400 (4) #-428 (4) #-438 (4) #-461 (4)
			#-479 (4)
E_READ	000000B0-R	434 (4)	#-409 (4)
E_REWIND	000000A1-R	425 (4)	#-411 (4)
E_SKIPFILE	000000D9-R	452 (4)	#-413 (4)
E_SKIPRECORD	000000FB-R	470 (4)	#-415 (4)
FCC_CPE	=00000001	208 (2)	
FCC_IDF	=00000000	207 (2)	
FCC_LAP	=00000003	210 (2)	
FCC_UPE	=00000002	209 (2)	
FUNC	=00000010	359 (3)	#-367 (4) #-408 (4) #-410 (4) #-412 (4)
			#-414 (4) #-417 (4)
HC_DRI	=0000C00B	256 (2)	
HC_GST	=0000C00F	257 (2)	
HC_NOP	=00000000	240 (2)	241 (2) 242 (2) 243 (2) 244 (2)
			245 (2)
HC_PAK	=00000000	241 (2)	
HC_RDN	=0000C001	248 (2)	#-435 (4)
HC_RPS	=00000000	244 (2)	
HC_RWD	=0000C408	254 (2)	#-426 (4)
HC_SCH	=00000000	245 (2)	
HC_SRF	=0000C008	250 (2)	#-472 (4)
HC_SRR	=0000C108	251 (2)	#-476 (4)
HC_STF	=0000C208	252 (2)	#-454 (4)
HC_STR	=0000C308	253 (2)	#-458 (4)
HC_UNL	=0000C10A	255 (2)	
HC_WCK	=00000000	242 (2)	
HC_WKR	=00000000	243 (2)	
HC_WRC	=0000C004	249 (2)	#-394 (4)
IO$_DRVCLR	=00000004		#-367 (4) #-417 (4)
IO$_READLBLK	=00000021		#-408 (4)
IO$_REWIND	=00000024		#-410 (4)
IO$_SKIPFILE	=00000025		#-412 (4)
IO$_SKIPRECORD	=00000026		#-414 (4)
LBN	=0000000C	358 (3)	
MAP_VA	0000018F-R	557 (4)	#-388 (4) #-396 (4) #-544 (4)
MODE	=00000014	360 (3)	
MS$CMD	00000000-R	291 (2)	#-377 (4) #-558 (4)

```
MSG_ATN                         =00000013      219     (2)
MSG_END                         =00000010      216     (2)
MSG_ERR                         =00000012      218     (2)
MSG_FAL                         =00000011      217     (2)
MSG_LOG                         =00000014      220     (2)
MS_BACT                          00000002                      #-397     (4)     #-436     (4)     #-455     (4)     #-459     (4)
                                                               #-473     (4)     #-477     (4)
MS_CHWD                          0000000E                      #-392     (4)
MS_CNT                           00000006                      #-398     (4)     #-437     (4)
MS_CPHD                          00000000                      #-394     (4)     #-426     (4)     #-435     (4)     #-454     (4)
                                                               #-458     (4)     #-472     (4)     #-476     (4)
MS_CPHD_M_ACK                   =00008000                        248     (2)       249     (2)       250     (2)       251     (2)
                                                                 252     (2)       253     (2)       254     (2)       255     (2)
                                                                 256     (2)       257     (2)
MS_CPHD_M_CVC                   =00004C00                        248     (2)       249     (2)       250     (2)       251     (2)
                                                                 252     (2)       253     (2)       254     (2)       255     (2)
                                                                 256     (2)       257     (2)
MS_LNTH                          0000000C                      #-391     (4)
MS_MBAO                          00000008                      #-390     (4)       395     (4)
MS_MHD                           00000010                      #-386     (4)       387     (4)
MS_MHD_M_ACK                    =00008000                      #-386     (4)
MS_RBPC                          00000014                      #-440     (4)
MS_TSSR_M_A16                   =00000100      284     (2)
MS_TSSR_M_A17                   =00000200      284     (2)
MS_TSSR_M_NBA                   =00000400      284     (2)     #-499     (4)
MS_TSSR_M_NXM                   =00000800      284     (2)     #-498     (4)
MS_TSSR_M_OFL                   =00000040      284     (2)
MS_TSSR_M_RMR                   =00001000      284     (2)     #-498     (4)
MS_TSSR_M_SC                    =00008000      284     (2)
MS_TSSR_M_SPE                   =00002000      284     (2)     #-497     (4)
MS_TSSR_M_SSR                   =00000080      284     (2)     #-490     (4)
MS_TSSR_M_UPE                   =00004000      284     (2)     #-497     (4)
MS_TSSR_S_FC                    =00000002      284     (2)
MS_TSSR_S_TCC                   =00000003      284     (2)     #-505     (4)
MS_TSSR_V_A16                   =00000008      284     (2)
MS_TSSR_V_A17                   =00000009      284     (2)
MS_TSSR_V_FC                    =00000004      284     (2)
MS_TSSR_V_NBA                   =0000000A      284     (2)     #-366     (4)
MS_TSSR_V_NXM                   =0000000B      284     (2)
MS_TSSR_V_OFL                   =00000006      284     (2)
MS_TSSR_V_RMR                   =0000000C      284     (2)
MS_TSSR_V_SC                    =0000000F      284     (2)
MS_TSSR_V_SPE                   =0000000D      284     (2)
MS_TSSR_V_SSR                   =00000007      284     (2)
MS_TSSR_V_TCC                   =00000001      284     (2)     #-505     (4)
MS_TSSR_V_UPE                   =0000000E      284     (2)
MS_XSRO                          00000016                        443     (4)     #-517     (4)       528     (4)
MS_XSRO_M_EOT                   =00000002                      #-515     (4)
MS_XSRO_M_EOT                   =00000001                      #-516     (4)
MS_XSRO_M_TMK                   =00008000                      #-514     (4)
MS_XSRO_V_ONL                   =00000006                      #-527     (4)
MS_XSRO_V_RLL                   =0000000C                      #-443     (4)
PTE$C_KW                        =10000000                        381     (4)
PTE$M_VALID                     =80000000                        381     (4)
RETURN                           000000A0-R     420     (4)     #-402     (4)     #-418     (4)
SIZ...                          =00000001      284     (2)       284     (2)
SIZE                            =00000008      357     (3)
```

```
SS$_ABORT                       =0000002C                             #-578    (4)
SS$_CTRLERR                     =00000054                             #-490    (4)     #-508    (4)     #-530    (4)
SS$_DATAOVERUN                  =00000838                             #-445    (4)
SS$_DEVOFFLINE                  =00000084                             #-526    (4)
SS$_ENDOFFILE                   =00000870                             #-519    (4)
SS$_ILLIOFUNC                   =000000F4                             #-419    (4)
SS$_NORMAL                      =00000001                             #-416    (4)     #-453    (4)     #-471    (4)     #-535    (4)
SS$_TAPEPOSLOST                 =00000224                             #-369    (4)
SS$_TIMEOUT                     =0000022C                             #-493    (4)
TCC_ATN                         =00000001       195     (2)
TCC_FNR                         =00000003       197     (2)
TCC_FTL                         =00000007       201     (2)
TCC_NML                         =00000000       194     (2)     #-507    (4)
TCC_REM                         =00000004       198     (2)
TCC_REN                         =00000C05       199     (2)
TCC_TSA                         =00000002       196     (2)
TCC_UER                         =00000006       200     (2)
TS11_DRIVER                      00000000-R     361.1   (4)       302    (2)       586    (4)
TS11_DRVSIZ                     =000001D2       586     (4)       302    (2)
TSBA                            =00000000       268     (2)     #-546    (4)
TSDB                            =00000000       267     (2)
TSSR                            =00000002       269     (2)     #-365    (4)     #-490    (4)     #-502    (4)
VA$S_BYTE                       =00000009                       #-372    (4)
VA$S_VPN                        =00000015                       #-378    (4)     #-561    (4)     #-564    (4)     #-569    (4)
                                                                #-572    (4)
VA$V_BYTE                       =00000000                       #-372    (4)
VA$V_VPN                        =00000009                       #-378    (4)     #-561    (4)     #-564    (4)     #-569    (4)
                                                                #-572    (4)
WAIT                             0000011D-R     488     (4)     #-401    (4)     #-429    (4)     #-439    (4)     #-463    (4)
                                                                #-480    (4)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	297.1 (2)	297.1 (2)
$BTDDEF	1	42 (2)	42 (2)
$DEFINI	1	42 (2)	42 (2) 43 (2) 44 (2) 45 (2) 46 (2) 47 (2) 48 (2) 49 (2) 71 (2)
$IODEF	17	43 (2)	43 (2)
$NDTDEF	2	44 (2)	44 (2)
$PRDEF	4	46 (2)	46 (2)
$PTEDEF	3	45 (2)	45 (2)
$RFBDEF	5	47 (2)	47 (2)
$SSDEF	21	48 (2)	48 (2)
$VADEF	1	49 (2)	49 (2)
$VIELD1	1		284 (2)
CASE	1	507 (4)	507 (4)
_VIELD	1		270.1 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.15	00:00:00.82
Command processing	158	00:00:00.85	00:00:03.37
Pass 1	617	00:00:15.24	00:00:52.92
Symbol table sort	[illegible]	00:00:01.94	00:00:03.89
Pass 2	160	00:00:03.35	00:00:06.77
Symbol table output	16	00:00:00.11	00:00:00.29
Psect synopsis output	7	00:00:00.04	00:00:00.06
Cross-reference output	37	00:00:00.53	00:00:01.21
Assembler run totals	1015	00:00:22.22	00:01:09.34

The working set limit was 1000 pages.
71[illegible] bytes (141 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1284 non-local and 17 local symbols.
602 source lines were read in Pass 1, producing 0 object records in Pass 2.
51 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	1
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	6
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	14

1185 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) TSBTDRIVR/UPDA=(TSBTDRIVR.UPD,TSBTDRIVR.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[S
```

Table of contents

(1) 60 UNWIND PROCEDURE CALL STACK
(1) 206 UNWIND CONDITION HANDLER
(1) 221 CALCULATE VALUE OF SP BEFORE CALL

```
                        0000     1          .TITLE  UNWIND - SYSTEM SERVICE UNWIND PROCEDURE CALL STACK
                        0000     2          .IDENT  /01-1/
                        0000     3
                        0000     4 ;
                        0000     5 ; COPYRIGHT (C) 1977, 1980
                        0000     6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        0000     7 ;
                        0000     8 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
                        0000     9 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
                        0000    10 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
                        0000    11 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
                        0000    12 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
                        0000    13 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
                        0000    14 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
                        0000    15 ;
                        0000    16 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
                        0000    17 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
                        0000    18 ; EQUIPMENT CORPORATION.
                        0000    19 ;
                        0000    20 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                        0000    21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    22 ;
                        0000    23 ; D. N. CUTLER 16-DEC-76
                        0000    24 ;
                        0000    25 ; R. S. Riggs  9-April-1979
                        0000    26 ;       01      Added REI to STARTUNWIND to return to correct stack
                        0000    27 ;               after exception.
                        0000    28 ;
                        0000    29 ; SYSTEM SERVICE UNWIND PROCEDURE CALL STACK
                        0000    30 ;
                        0000    31 ; MACRO LIBRARY CALLS
                        0000    32 ;
                        0000    33
                        0000    34          $CHFDEF                             ;DEFINE CONDITION HANDLING ARGLIST OFFSETS
                        0000    35          $SSDEF                              ;DEFINE SYSTEM STATUS VALUES
                        0000    36
                        0000    37 ;
                        0000    38 ; LOCAL SYMBOLS
                        0000    39 ;
                        0000    40 ; ARGUMENT LIST OFFSET DEFINITIONS
                        0000    41 ;
                        0000    42
00000004                0000    43 DEPADR=4                                    ;ADDRESS OF NUMBER OF FRAMES TO UNWIND
00000008                0000    44 NEWPC=8                                     ;CHANGE OF FLOW FINAL RETURN ADDRESS
                        0000    45
                        0000    46 ;
                        0000    47 ; CALL FRAME OFFSET DEFINITIONS
                        0000    48 ;
                        0000    49
00000000                0000    50 HANDLER=0                                   ;CONDITION HANDLER ADDRESS
00000004                0000    51 SAVPSW=4                                    ;SAVED PSW FROM CALL
00000006                0000    52 SAVMSK=6                                    ;REGISTER SAVE MASK
00000008                0000    53 SAVAP=8                                     ;SAVED AP REGISTER IMAGE
0000000C                0000    54 SAVFP=12                                    ;SAVED FP REGISTER IMAGE
00000010                0000    55 SAVPC=16                                    ;SAVED PC REGISTER IMAGE
00000014                0000    56 SAVRG=20                                    ;OTHER SAVED REGISTER IMAGES
                        0000    57
```

```
                        00000000     58          .PSECT  SEP, SHR, WRT, EXE, LONG
```

```
                                0000     60          .SBTTL  UNWIND PROCEDURE CALL STACK
                                0000     61 ;+
                                0000     62 ; EXE$UNWIND - UNWIND PROCEDURE CALL STACK
                                0000     63 ;
                                0000     64 ; THIS SERVICE PROVIDES THE CAPABILITY TO UNWIND THE PROCEDURE CALL STACK
                                0000     65 ; TO A SPECIFIED DEPTH AFTER A HARDWARE- OR SOFTWARE-DETECTED EXCEPTION
                                0000     66 ; CONDITION HAS BEEN SIGNALLED. OPTIONALLY A CHANGE OF FLOW RETURN ADDRESS
                                0000     67 ; MAY ALSO BE SPECIFIED. THE ACTUAL UNWIND IS NOT PERFORMED IMMEDIATELY BY
                                0000     68 ; THE SERVICE, BUT RATHER THE RETURN ADDRESSES IN THE CALL STACK ARE MODIFIED
                                0000     69 ; SUCH THAT WHEN THE CONDITION HANDLER RETURNS THE UNWIND OCCURS.
                                0000     70 ;
                                0000     71 ; INPUTS:
                                0000     72 ;
                                0000     73 ;       DEPADR(AP) = ADDRESS OF NUMBER OF FRAMES TO UNWIND.
                                0000     74 ;       NEWPC(AP) = CHANGE OF FLOW FINAL RETURN ADDRESS.
                                0000     75 ;
                                0000     76 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                0000     77 ;
                                0000     78 ; OUTPUTS:
                                0000     79 ;
                                0000     80 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FULLY UNWIND CALL STACK.
                                0000     81 ;
                                0000     82 ;               R0 = SS$_ACCVIO - CALL STACK NOT ACCESSIBLE TO CALLING ACCESS
                                0000     83 ;                    MODE.
                                0000     84 ;
                                0000     85 ;               R0 = SS$_INSFRAME - INSUFFICIENT CALL FRAMES TO UNWIND TO
                                0000     86 ;                    SPECIFIED DEPTH.
                                0000     87 ;
                                0000     88 ;               R0 = SS$_NOSIGNAL - NO SIGNAL IS CURRENTLY ACTIVE TO UNWIND.
                                0000     89 ;
                                0000     90 ;               R0 = SS$_UNWINDING - UNWIND ALREADY IN PROGRESS.
                                0000     91 ;
                                0000     92 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0000     93 ;
                                0000     94 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0000     95 ;-
                                0000     96
                         003C   0000     97          .ENTRY  EXE$UNWIND,^M<R2,R3,R4,R5>
      6D    00EA'CF    DE     0002     98          MOVAL   W^CHANDL,(FP)           ;ESTABLISH CONDITION HANDLER
            54    5D   D0     0007     99          MOVL    FP,R4                   ;SET ADDRESS OF FIRST FRAME TO EXAMINE
                               000A    100
                               000A    101 ;
                               000A    102 ; SEARCH CALL STACK FOR A FRAME THAT WAS CREATED BY A CALL FROM THE SIGNAL
                               000A    103 ; DISPATCH VECTOR OR BY A CALL FROM THE UNWIND SIGNAL DISPATCHER.
                               000A    104 ;
                               000A    105
      50    0900 8F    3C     000A    106          MOVZWL  #SS$_NOSIGNAL,R0        ;ASSUME NO SIGNAL ACTIVE
        54    0C A4    D0     000F    107 10$:     MOVL    SAVFP(R4),R4            ;GET ADDRESS OF PREVIOUS FRAME
                     19   13   0013    108          BEQL    20$                     ;IF EQL END OF CALL STACK
10 A4    0000000'  8F    D1   0015    109          CMPL    #SYS$CALL_HANDL+4,SAVPC(R4) ;CALL FROM CONDITION HANDLER DISPATCHER?
                     10   13   001D    110          BEQL    30$                     ;IF EQL YES
10 A4    000000D2' 8F    D1   001F    111          CMPL    #CALLUNWIND+4,SAVPC(R4) ;CALL FROM UNWIND SIGNAL DISPATCHER?
                     E6   12   0027    112          BNEQ    10$                     ;IF NEQ NO
      50    0928 8F    3C     0029    113          MOVZWL  #SS$_UNWINDING,R0       ;SET ALREADY UNWINDING
                     04   002E    114 20$:     RET                             ;
                               002F    115
                               002F    116 ;
```

```
                                 002F    117 ; SET TO UNWIND PROCEDURE CALL STACK TO SPECIFIED DEPTH
                                 002F    118 ;
                                 002F    119
            53   04 AC    D0     002F    120 30$:    MOVL    DEPADR(AP),R3           ;GET ADDRESS OF NUMBER OF FRAMES TO UNWIND
                    05    13     0033    121         BEQL    40$                     ;IF EQL NONE SPECIFIED
                 53   63    D0     0035    122         MOVL    (R3),R3                 ;GET NUMBER OF FRAMES TO UNWIND
                    0F    11     0038    123         BRB     50$                     ;
                 52   54    D0     003A    124 40$:    MOVL    R4,R2                   ;COPY CURRENT FRAME ADDRESS
                      00C9  30     003D    125         BSBW    OLDSP                   ;CALCULATE VALUE OF SP BEFORE CALL
                 52   0C A2   D0     0040    126         MOVL    CHF$L_MCHARGLST+4(R2),R2 ;GET ADDRESS OF MECHANISM ARGUMENT LIST
            53   08 A2   01    C1     0044    127         ADDL3   #1,CHF$L_MCH_DEPTH(R2),R3 ;CALCULATE DEPTH OF ESTABLISHER'S CALLER
                    3E    15     0049    128 50$:    BLEQ    90$                     ;IF LEQ NO FRAMES TO REMOVE
              10 A4   A3'AF   DE     004B    129         MOVAL   B^STARTUNWIND,SAVPC(R4) ;SET CONDITION HANDLER UNWIND ADDRESS
                                 0050    130
                                 C050    131 ;
                                 0050    132 ; SCAN THROUGH SPECIFIED NUMBER OF FRAMES SETTING EACH FRAME TO UNWIND ON RETURN
                                 0050    133 ;
                                 0050    134
            55   0C A4   D0     0050    135 60$:    MOVL    SAVFP(R4),R5            ;GET ADDRESS OF PREVIOUS FRAME
                    47    13     0054    136         BEQL    100$                    ;IF EQL INSUFICIENT FRAMES
                    53    D7     0056    137         DECL    R3                      ;ANY MORE FRAMES TO CONSIDER?
                    2F    15     0058    138         BLEQ    90$                     ;IF LEQ NO
          10 A5   00000004'8F   D1     005A    139         CMPL    #SYS$CALL_HANDL+4,SAVPC(R5) ;CALL FROM CONDITION DISPATCHER?
                    18    12     0062    140         BNEQ    80$                     ;IF NEQ NO
                 52   55    D0     0064    141         MOVL    R5,R2                   ;COPY ADDRESS OF CURRENT FRAME
                      009F  30     0067    142         BSBW    OLDSP                   ;CALCULATE VALUE OF SP BEFORE CALL
              52   0C A2   D0     006A    143         MOVL    CHF$L_MCHARGLST+4(R2),R2 ;GET ADDRESS OF MECHANISM ARGUMENT LIST
              04 A2   55    D1     006E    144 70$:    CMPL    R5,CHF$L_MCH_FRAME(R2)  ;UNWOUND TO ESTABLISHER FRAME?
                    0B    13     0072    145         BEQL    80$                     ;IF EQL YES
          10 A5   B6'AF   DE     0074    146         MOVAL   B^LOOPUNWIND,SAVPC(R5)  ;SET FRAME UNWIND ADDRESS
              55   0C A5   D0     0079    147         MOVL    SAVFP(R5),R5            ;GET ADDRESS OF PREVIOUS FRAME
                    EF    11     007D    148         BRB     70$                     ;
          10 A5   B6'AF   DE     007F    149 80$:    MOVAL   B^LOOPUNWIND,SAVPC(R5)  ;SET FRAME UNWIND ADDRESS
                 54   55    D0     0084    150         MOVL    R5,R4                   ;SET ADDRESS OF CURRENT FRAME
                    C7    11     0087    151         BRB     60$                     ;
                                 0089    152
                                 0089    153 ;
                                 0089    154 ; MODIFY CHANGE OF FLOW RETURN IF NEW ADDRESS SPECIFIED
                                 0089    155 ;
                                 0089    156
                 50   01    3C     0089    157 90$:    MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION
              51   08 AC   D0     008C    158         MOVL    NEWPC(AP),R1            ;GET CHANGE OF FLOW RETURN ADDRESS
                    10    13     0090    159         BEQL    110$                    ;IF EQL NONE SPECIFIED
              55   0C A4   D0     0092    160         MOVL    SAVFP(R4),R5            ;GET ADDRESS OF PREVIOUS FRAME
                    05    13     0096    161         BEQL    100$                    ;IF EQL INSUFFICIENT FRAMES
          10 A5   51    D0     0098    162         MOVL    R1,SAVPC(R5)            ;SET NEW FINAL RETURN ADDRESS
                    04    009C    163         RET                             ;
           50   012C 8F   3C     009D    164 100$:   MOVZWL  #SS$_INSFRAME,R0        ;SET INSUFFICIENT FRAMES
                    04    00A2    165 110$:   RET                             ;
                                 00A3    166
                                 00A3    167 ;
                                 00A3    168 ; UNWIND HANDLER FRAME
                                 00A3    169 ;
                                 00A3    170
                                 00A3    171 STARTUNWIND:                            ;START OF ACTUAL UNWIND
        6E   24 AE   09    C1     00A3    172         ADDL3   #9,36(SP),(SP)          ; Count of longwords to PSL
                 6E   04    C4     00A8    173         MULL2   #4,(SP)                 ; Byte offset
```

```
                50   1C AE    7D  00AB   174          MOVQ    28(SP),R0                       ; Restore R0/R1
                   5E   6E    C0  00AF   175          ADDL2   (SP),SP                         ; Remove all args down to PSL
                     B6'AF    9F  00B2   176          PUSHAB  B^LOOPUNWIND                    ; Address to resume on correct stack
                              02  00B5   177          REI                                     ; Resume below on correct stack
                                  00B6   178
                                  00B6   179 ;
                                  00B6   180 ; UNWIND CALL FRAME SIGNALING CONDITION HANDLER IF ONE IS SPECIFIED
                                  00B6   181 ;
                                  00B6   182
                                  00B6   183          .ENABL  LSB
                                  00B6   184 LOOPUNWIND:                                      ;UNWIND CALL FRAME
                         6D       D5  00B6   185          TSTL    (FP)                        ;CONDITION HANDLER SPECIFIED?
                         1C       13  00B8   186          BEQL    10$                         ;IF EQL NO
                7E    0920 8F       3C  00BA   187          MOVZWL  #SS$_UNWIND,-(SP)           ;PUSH UNWIND SIGNAL CONDITION
                         01       DD  00BF   188          PUSHL   #1                          ;PUSH NUMBER OF SIGNAL ARGUMENTS
                         03       BB  00C1   189          PUSHR   #^M<R0,R1>                  ;PUSH REGISTERS R0 AND R1
                         00       DD  00C3   190          PUSHL   #0                          ;PUSH FRAME DEPTH
                         5D       DD  00C5   191          PUSHL   FP                          ;PUSH FRAME ADDRESS
                         04       DD  00C7   192          PUSHL   #4                          ;PUSH NUMBER OF MECHANISM ARGUMENTS
                         6E       DF  00C9   193          PUSHAL  (SP)                        ;PUSH ADDRESS OF MECHANISM ARGUMENTS
                      18 AE       DF  00CB   194          PUSHAL  24(SP)                      ;PUSH ADDRESS OF SIGNAL ARGUMENTS
                                  00CE   195 CALLUNWIND:                                      ;SIGNAL UNWIND
                00 BD    02       FB  00CE   196          CALLS   #2,@(FP)                    ;CALL CONDITION HANDLER
                50    0C AE       7D  00D2   197          MOVQ    CHF$L_MCH_SAVR0(SP),R0      ;RETRIEVE NEW VALUES FOR R0 AND R1
                5C    14 AD       DE  00D6   198 10$:     MOVAL   SAVR0(FP),AP                ;GET ADDRESS OF REGISTER SAVE AREA
                   03 06 AD       E9  00DA   199          BLBC    SAVMSK(FP),20$              ;IF LBC R0 NOT SAVED
                   8C    50       D0  00DE   200          MOVL    R0,(AP)+                    ;SAVE R0 FOR SUBSEQUENT RESTORATION
             05 06 AD    01       E1  00E1   201 20$:     BBC     #1,SAVMSK(FP),30$           ;IF CLR, R1 NOT SAVED
                   6C    51       D0  00E6   202          MOVL    R1,(AP)                     ;SAVE R1 FOR SUBSEQUENT RESTORATION
                              04   00E9   203 30$:     RET                                     ;
                                  00EA   204          .DSABL  LSB
```

```
                                00EA   206          .SBTTL  UNWIND CONDITION HANDLER
                                00EA   207 ;
                                00EA   208 ; UNWI [ CONDITION HANDLER
                                00EA   209 ;
                                00EA   210
                          0000  00EA   211 CHANDL: .WORD   0                                ;ENTRY MASK
          50    04 AC     D0   00EC   212          MOVL    CHF$L_SIGARGLST(AP),R0         ;GET ADDRESS OF SIGNAL ARGUMENT LIST
 04 A0    0920  8F     B1   00F0   213          CMPW    #SS$_UNWIND,CHF$L_SIG_NAME(R0) ;UNWINDING?
                10     13   00F6   214          BEQL    10$                            ;IF EQL YES
          51    08 AC     D0   00F8   215          MOVL    CHF$L_MCHARGLST(AP),R1         ;GET ADDRESS OF MECHANISM ARGUMENT LIST
 0C A1    04 A0     D0   00FC   216          MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;SET FINAL RETURN STATUS
                7E     7C   0101   217          CLRQ    -(SP)                          ;CLEAR DEPTH AND NEW PC ARGUMENTS
    FEF8 CF     02     FB   0103   218          CALLS   #2,W^EXE$UNWIND                ;UNWIND TO ESTABLISHER'S CALLER
                       04   0108   219 10$:     RET                                    ;
```

```
                                0109   221          .SBTTL  CALCULATE VALUE OF SP BEFORE CALL
                                0109   222 ;
                                0109   223 ; SUBROUTINE TO CALCULATE VALUE OF SP BEFORE CALL
                                0109   224 ;
                                0109   225
7E   06 A2   02   0E   EF       0109   226 OLDSP:  EXTZV   #14,#2,SAVMSK(R2),-(SP) ;GET STACK ALIGNMENT BIAS
51   06 A2   0C   00   EF       010F   227          EXTZV   #0,#12,SAVMSK(R2),R1    ;GET REGISTER SAVE MASK
                52   14   C0    0115   228          ADDL    #SAVRG,R2               ;ADD OFFSET TO REGISTER SAVE AREA
                52   8E   C0    0118   229          ADDL    (SP)+,R2                ;ADD STACK ALIGNMENT BIAS
                  03 51   E9    011B   230 10$:     BLBC    R1,20$                  ;IF LBC CORRESPONDING REGISTER NOT SAVED
                52   04   C0    011E   231          ADDL    #4,R2                   ;ADJUST FOR SAVED REGISTER
      51   51   FF 8F   78    0121   232 20$:     ASHL    #-1,R1,R1               ;ANY MORE REGISTERS SAVED?
                     F3   12    0126   233          BNEQ    10$                     ;IF NEQ YES
                          05    0128   234          RSB                             ;
                                0129   235
                                0129   236          .END
```

B 1 RMS$DISCONNECT routine
C 1 RMS$CLOSE routine
D 1 RMS$CLOSE routine
E 1 RMS$CLOSE routine
F 1 *** RT11 Console media RMS rou
G 1 declarations
H 1 advance record pointers
I 1 Access file block via cache
J 1 Access file block via cache
K 1 Access file block via cache
L 1 RT-11 OPEN service
M 1 RT-11 OPEN service
N 1 RT-11 OPEN service
B 2 RT-11 OPEN service
C 2 RT11$GET routine
D 2 RT11$GET routine
E 2 RT11$GET routine
F 2 RT11$GET routine
G 2 RT11$GET routine
H 2 RT11$READ routine
I 2 RT11$READ routine
J 2 RT11$READ routine
K 2 SYSTEM CONTROL BLOCK
L 2 SYSTEM CONTROL BLOCK
M 2 SYSTEM CONTROL BLOCK
N 2 SYSTEM CONTROL BLOCK
B 3 DECLARATIONS
C 3 DECLARATIONS
D 3 DECLARATIONS
E 3 INITIAL SYSTEM CONTROL BLOCK I
F 3 INITIAL SYSTEM CONTROL BLOCK I
G 3 DSX$INITSCB INITIALIZE SYSTEM
H 3 DSX$INITSCB INITIALIZE SYSTEM
I 3 DSX$INITSCB INITIALIZE SYSTEM
J 3 DSX$SETVEC SET VECTOR ROUTINE
K 3 DSX$SETVEC SET VECTOR ROUTINE
L 3 DSX$SETVEC SET VECTOR ROUTINE
M 3 DSX$CLRVEC RESTORE VECTOR ROUT
N 3 DSX$CLRVEC RESTORE VECTOR ROUT
B 4 CHECK_VECTOR Verify vector ran
C 4 CHECK_VECTOR Verify vector ran
D 4 DSX$SETIPL SET INTERRUPT PRIOR
E 4 DSX$SETIPL SET INTERRUPT PRIOR
F 4 SOFT$INT Software interrupt ve
G 4 Symbol table
H 4 Symbol table
I 4 Symbol table
J 4 Psect synopsis
K 4 Cross reference
L 4 Cross reference
M 4 Cross reference
N 4 Cross reference
B 5 Cross reference
C 5 Cross reference
D 5 Cross reference
E 5 Cross reference
F 5 *** SCRIPT Read from command s
G 5 *** SCRIPT Read from command s
H 5 *** SCRIPT Read from command s
I 5 Libraries and External Symbols
J 5 External Symbol Declarations
K 5 Macros and Equated Symbols
L 5 Data Psect Declarations
M 5 Work Psect Declarations
N 5 Absolute Psect Declarations
B 6 SCRIPT$INIT - Initialize scrip
C 6 SCRIPT$INIT - Initialize scrip
D 6 SCRIPT$OPEN - Initialize a scr
E 6 SCRIPT$OPEN - Initialize a scr
F 6 SCRIPT$OPEN - Initialize a scr
G 6 SCRIPT$OPEN - Initialize a scr
H 6 SCRIPT$OPEN - Initialize a scr
I 6 SCRIPT$FLUSH, STOP, CONT - Ter
J 6 SCRIPT$FLUSH, STOP, CONT - Ter
K 6 SCRIPT$FINISH - Terminate scri
L 6 SCRIPT$FINISH - Terminate scri
M 6 DS_GETLINE - Get program data
N 6 DS_GETLINE - Get program data
B 7 DS_GETLINE - Get program data
C 7 DS_GETLINE - Get program data
D 7 DS_GETLINE - Get program data
E 7 DS_GETLINE - Get program data
F 7 DS_GETLINE - Get program data
G 7 DS_GETLINE - Get program data
H 7 DS_GETLINE - Get program data
I 7 DS_GETLINE - Get program data
J 7 DS_GETLINE - Get program data
K 7 DS_GETLINE - Get program data
L 7 COPY_IT - Copy and translate
M 7 COPY_IT - Copy and translate
N 7 ADVANCE - Advance to next reco
B 8 ADVANCE - Advance to next reco
C 8 Symbol table
D 8 Symbol table
E 8 Symbol table
F 8 Symbol table
G 8 Cross reference
H 8 Cross reference
I 8 Cross reference
J 8 Cross reference
K 8 Cross reference
L 8 Cross reference
M 8 Cross reference
N 8 Cross reference
B 9 Cross reference
C 9 *** SHOWMEM Handle Show Memory
D 9 *** SHOWMEM Handle Show Memory
E 9 *** SHOWMEM Handle Show Memory
F 9 *** SHOWMEM Handle Show Memory
G 9 Is a program loaded, really?
H 9 Is a program loaded, really?
I 9 Is a program loaded, really?
J 9 DSV$ShowMemory routine
K 9 ShowMemory Routine
L 9 ShowMemory Routine
M 9 ShowMemory Routine
N 9 ShowMemory Routine
B 10 ShowMemory Routine
C 10 ShowMemory Routine
D 10 ShowMemory Routine
E 10 ShowMemory Routine
F 10 ShowMemory Routine
G 10 ShowMemory Routine
H 10 ShowMemory Routine
I 10 ShowMemory Routine
J 10 ShowMemory Routine
K 10 VRSETMEM
L 10 VRSETMEM
M 10 *** START Handle START command
N 10 *** START Handle START command
B 11 *** START Handle START command
C 11 *** START Handle START command
D 11 *** START Handle START command
E 11 Declarations
F 11 Declarations
G 11 Declarations
H 11 Declarations
I 11 VRStart and VRReStart Routines
J 11 VRStart and VRReStart Routines
K 11 VRStart and VRReStart Routines
L 11 VRStart and VRReStart Routines
M 11 VRStart and VRReStart Routines
N 11 VRStart and VRReStart Routines
B 12 VRStart and VRReStart Routines
C 12 VRStart and VRReStart Routines
D 12 VRStart and VRReStart Routines
E 12 VRStart and VRReStart Routines
F 12 VRStart and VRReStart Routines
G 12 VRStart and VRReStart Routines
H 12 VRStart and VRReStart Routines
I 12 VRStart and VRReStart Routines
J 12 VRStart and VRReStart Routines
K 12 VRStart and VRReStart Routines
L 12 VRAbort and DSX$Abort Routines
M 12 VRAbort and DSX$Abort Routines
N 12 VRAbort and DSX$Abort Routines
B 13 VRAbort and DSX$Abort Routines
C 13 VRSupport Routines
D 13 VRSupport Routines
E 13 VRShowSections Routine
F 13 VRShowSections Routine
G 13 DSI$TIME_AST Routine
H 13 Symbol table
I 13 Symbol table
J 13 Symbol table
K 13 Symbol table
L 13 Psect synopsis
M 13 Cross reference
N 13 Cross reference
B 14 Cross reference
C 14 Cross reference
D 14 Cross reference
E 14 Cross reference
F 14 Cross reference
G 14 Cross reference
H 14 Cross reference
I 14 Cross reference
J 14 *** TSBTDRIVR driver for TS11/
K 14 *** TSBTDRIVR driver for TS11/
L 14 DECLARATIONS

M 14 DECLARATIONS
N 14 DECLARATIONS
5 15 DECLARATIONS
C 15 DECLARATIONS
D 15 TS11/TS04, TU80 Bootstrap driv
E 15 TS11/TS04, TU80 Bootstrap driv
F 15 TS11/TS04, TU80 Bootstrap driv
G 15 TS11/TS04, TU80 Bootstrap driv
H 15 TS11/TS04, TU80 Bootstrap driv
I 15 TS11/TS04, TU80 Bootstrap driv
J 15 TS11/TS04, TU80 Bootstrap driv
K 15 Symbol table
L 15 Symbol table
M 15 Cross reference
N 15 Cross reference
B 16 Cross reference
C 16 Cross reference
D 16 Cross reference
E 16 - SYSTEM SERVICE UNWIND PROCED
F 16 - SYSTEM SERVICE UNWIND PROCED
G 16 - SYSTEM SERVICE UNWIND PROCED
H 16 UNWIND PROCEDURE CALL STACK
I 16 UNWIND PROCEDURE CALL STACK
J 16 UNWIND PROCEDURE CALL STACK
K 16 UNWIND CONDITION HANDLER
L 16 CALCULATE VALUE OF SP BEFORE C

```
CALLUNWIND                      000000CE R   D   02
CHANDL                          000000EA R   D   02
CHF$L_MCHARGLST             =   00000008     D
CHF$L_MCH_DEPTH             =   00000008     D
CHF$L_MCH_FRAME             =   00000004     D
CHF$L_MCH_SAVR0             =   0000000C     D
CHF$L_SIGARGLST             =   00000004     D
CHF$L_SIG_NAME              =   00000004     D
DEPADR                      =   00000004     D
EXE$UNWIND                      00000000 RG  D   02
HANDLER                     =   00000000     D
LOOPUNWIND                      000000B6 R   D   02
NEWPC                       =   00000008     D
OLDSP                           00000109 R   D   02
SAVAP                       =   00000008     D
SAVFP                       =   0000000C     D
SAVMSK                      =   00000006     D
SAVPC                       =   00000010     D
SAVPSW                      =   00000004     D
SAVRG                       =   00000014     D
SS$_INSFRAME                =   0000012C     D
SS$_NORMAL                  =   00000001     D
SS$_NOSIGNAL                =   00000900     D
SS$_UNWIND                  =   00000920     D
SS$_UNWINDING               =   00000928     D
STARTUNWIND                     000000A3 R   D   02
SYS$CALL_HANDL                  ********   X   02
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000129 (297.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
CALLUNWIND	000000CE-R	195 (1)	#-111 (1)
CHANDL	000000EA-R	211 (1)	98 (1)
CHF$L_MCHARGLST	=00000008		#-126 (1) #-143 (1) #-215 (1)
CHF$L_MCH_DEPTH	=00000008		#-127 (1)
CHF$L_MCH_FRAME	=00000004		#-144 (1)
CHF$L_MCH_SAVR0	=0000000C		#-197 (1) #-216 (1)
CHF$L_SIGARGLST	=00000004		#-212 (1)
CHF$L_SIG_NAME	=00000004		#-213 (1) #-216 (1)
DEPADR	=00000004	43 (1)	#-120 (1)
EXE$UNWIND	00000000-R	97 (1)	218 (1)
HANDLER	=00000000	50 (1)	
LOOPUNWIND	000000B6-R	184 (1)	146 (1) 149 (1) 176 (1)
NEWPC	=00000008	44 (1)	#-158 (1)
OLDSP	00000109-R	226 (1)	#-125 (1) #-142 (1)
SAVAP	=00000008	53 (1)	
SAVFP	=0000000C	54 (1)	#-107 (1) #-135 (1) #-147 (1) #-160 (1)
SAVMSK	=00000006	52 (1)	#-199 (1) 201 (1) 226 (1) 227 (1)
SAVPC	=00000010	55 (1)	#-109 (1) #-111 (1) #-129 (1) #-139 (1)
			#-146 (1) #-149 (1) #-162 (1)
SAVPSW	=00000004	51 (1)	
SAVRG	=00000014	56 (1)	198 (1) #-228 (1)
SS$_INSFRAME	=0000012C		#-164 (1)
SS$_NORMAL	=00000001		#-157 (1)
SS$_NOSIGNAL	=00000900		#-106 (1)
SS$_UNWIND	=00000920		#-187 (1) #-213 (1)
SS$_UNWINDING	=00000928		#-113 (1)
STARTUNWIND	000000A3-R	171 (1)	129 (1)
SYS$CALL_HANDL	00000000-XR		#-109 (1) #-139 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...			
$CHFDEF	1	34 (1)	34 (1)			
$DEFINI	1	34 (1)	34 (1)	35 (1)		
$SSDEF	21	35 (1)	35 (1)			

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.12	00:00:00.25
Command processing	134	00:00:00.87	00:00:03.06
Pass 1	286	00:00:04.71	00:00:13.39
Symbol table sort	0	00:00:00.64	00:00:01.04
Pass 2	57	00:00:00.90	00:00:01.11
Symbol table output	5	00:00:00.05	00:00:00.05
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	15	00:00:00.13	00:00:00.16
Assembler run totals	539	00:00:07.46	00:00:19.09

The working set limit was 1000 pages.
23413 bytes (46 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 431 non-local and 17 local symbols.
236 source lines were read in Pass 1, producing 0 object records in Pass 2.
14 pages of virtual memory were used to define 8 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	5

481 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) UNWIND/UPDA=(UNWIND.UPD,UNWIND.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 36 DECLARATIONS

```
                        .title  ver730
                .ident  / 7.0/
-2                      .LIST   MEB
                        .NLIST  CND

                ;
                ; Copyright (c) 1977, 1982
                ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                ;
                ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                ; REMAIN IN DEC.
                ;
                ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                ; CORPORATION.
                ;
                ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                ;

                ;++
                ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
                ;
                ; ABSTRACT:
                ;
                ; ENVIRONMENT:
                ;
                ; AUTHOR:       KEN CHAPMAN     13-SEP-78       VERSION 01.
                ;--
```

```
                                0000     36          .SBTTL  DECLARATIONS
                                0000     37 ;
                                0000     38 ; INCLUDE FILES:
                                0000     39 ;
                                0000     40
                                0000     41 ;
                                0000     42 ; MACROS:
                                0000     43 ;
                                0000     44
                                0000     45 ;
                                0000     46 ; EQUATED SYMBOLS:
                                0000     47 ;
                                0000     48
                                0000     49
                                0000     50 ;
                                0000     51 ; OWN STORAGE:
                                0000     52 ;
                            00000000     53          .PSECT  DATA, SHR, NOEXE, NOWRT, LONG
                                0000     54 DS$GQ_DAYTIM::
                00000008'010E0000'  0000 .1 .ascid  /26-JUL-1984 09:25:07.15/
                4C 55 4A 2D 36 32   0008
                20 34 38 39 31 2D   000E
                3A 35 32 3A 39 30   0014
                   35 31 2E 37 30   001A
-1                                  001F     56 DS$GT_START::
                              34'   001F     57          .BYTE   10$-.-1
                47 41 49 44 2F 21   0020 .1         .ascii  '!/DIAGNOSTIC SUPERVISOR.  ZZ-ENSAA-'
                43 49 54 53 4F 4E   0026
                52 45 50 55 53 20   002C
                2E 52 4F 53 49 56   0032
                45 2D 5A 5A 20 20   0038
                   2D 41 41 53 4E   003E
                   2D 30 2E 37 20   0043 .2 .ascii  / 7.0-/
                         34 37 35   0048 .3 .ascii  /574/
-3              25 30 32 21 20 20   0048 61          .ASCII  '' !20%D!/''
                      2F 21 44      0051
                                0054     62 10$:
                                0054     63 DS$GT_VDS_Version::                          ;                                       [02]
                2D 30 2E 37 20 00'  0054 .1         .ASCIC  ' 7.0-574'
                         34 37 35   005A
                               08   0054
-1                                  005D     65 DS$GT_PROMPT::
                   20 3E 53 44 00'  005D     66          .ASCIC  \DS> \
                               04   005D
                                006A     67
                            00000000     68          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                                0000     69 DS$ENTRY::
                              0000'  0000     70          .VECTOR DS$USERMODE,^M<>
                 00000002'EF    17   0002     71          JMP     DS$USERMODE+2           ; GO TO USER MODE INIT CODE.    [02]
                                0008     72
                                0008     73          .END    DS$ENTRY
```

```
DS$ENTRY                        00000000 RG D  02
DS$GQ_DAYTIM                    00000000 RG D  01
DS$GT_PROMPT                    0000005D RG D  01
DS$GT_START                     0000001F RG D  01
DS$GT_VDS_VERSION               00000054 RG D  01
DS$USERMODE                     ******** RG D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DATA	00000062 (98.)	01 (1.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
DS$ENTRY	00000000-R	69 (1)	
DS$GQ_DAYTIM	00000000-R	54 (1)	
DS$GT_PROMPT	0000005D-R	65 (1)	
DS$GT_START	0000001F-R	56 (1)	
DS$GT_VDS_VERSION	00000054-R	63 (1)	
DS$USERMODE	00000000-XR		71 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.11	00:00:00.65
Command processing	142	00:00:00.83	00:00:02.84
Pass 1	102	00:00:00.67	00:00:01.81
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	54	00:00:00.38	00:00:01.72
Symbol table output	1	00:00:00.01	00:00:00.01
Psect synopsis output	4	00:00:00.03	00:00:00.15
Cross-reference output	4	00:00:00.03	00:00:00.03
Assembler run totals	341	00:00:02.06	00:00:07.21

The working set limit was 950 pages.
1101 bytes (3 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 6 non-local and 1 local symbols.
73 source lines were read in Pass 1, producing 0 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYSO.SYSMAINT]DS.MLB;218	0
DMA1:[SYSO.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) VER730/UPDA=(VER730.UPD,VER730.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYSO.SYSMA
```

Table of contents

(1) 39 History
(1) 49 Declarations
(1) 63 Return-only Dummy Routines ; [02]
(1) 108 MUTEX Control Routines

```
                        0000     1          .TITLE  VMSDUMMY        *** VMSDUMMY Fake VMS driver entry points
                        0000     2          .IDENT  '05-02'
                        0000     3
                        0000     4
                        0000     5 ;
                        0000     6 ;*****************************************************************************
                        0000     7 ;*                                                                           *
                        0000     8 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000     9 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000    10 ;*                                                                           *
                        0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000    12 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000    14 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000    16 ;*  TRANSFERRED.                                                            *
                        0000    17 ;*                                                                           *
                        0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                        0000    20 ;*  CORPORATION.                                                            *
                        0000    21 ;*                                                                           *
                        0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000    24 ;*                                                                           *
                        0000    25 ;*****************************************************************************
                        0000    26
                        0000    27 ;++
                        0000    28 ; FACILITY : Diagnostic Supervisor
                        0000    29 ;
                        0000    30 ; ABSTRACT:
                        0000    31 ;       This module includes dummies of VMS routines which are either
                        0000    32 ; unnecessary in the Supervisor, or exist only in drastically reduced form,
                        0000    33 ; and do not fit within any other Supervisor module.
                        0000    34 ;
                        0000    35 ; ENVIRONMENT:
                        0000    36 ;
                        0000    37 ;--
```

```
                        .Sbttl  History

;
; AUTHOR:               Dave Butenhof   CREATION DATE: 18-jun-1980
;
; MODIFIED BY:
;       01              Jack Stansbury, Version 6.8, July 1, 1982
;                       Took out the Exe$PwrTimChk routine label because I am adding
;                       the real routine to the QIO module.
;-
```

```
                        .Sbttl Declarations

;
; INCLUDE FILES
;

;
; EQUATES
;
        $SSDEF          ; Define SS return codes

;
; OWN STORAGE
;
```

```
                                        .Sbttl  Retur  only Dummy Routines      ;                                  [02]
00000000                                .Psect  Code  hr, Exe, NoWrt, Byte      ;                                  [02]
                                        ;++
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;       Routines which require no Supervisor action.  Rather than edit the
                                        ; calls from sources, these dummy entry points are provided to return
                                        ; immediately.
                                        ;
                                        ; CALLING SEQUENCE:
                                        ;       JSB type.
                                        ;
                                        ; INPUT PARAMETERS:
                                        ;       Ignored.
                                        ;
                                        ; IMPLICIT INPUTS:
                                        ;       None
                                        ;
                                        ; OUTPUT PARAMETERS:
                                        ;       R0 may indicate completion status or fake function value.
                                        ;
                                        ; IMPLICIT OUTPUTS:
                                        ;       none
                                        ;
                                        ; COMPLETION CODES:
                                        ;
                                        ; SIDE EFFECTS:
                                        ;       none
                                        ;--

                                        MT$CHECK_ACCESS::
50    D4                                        CLRL    R0
                                        EXE^SNDEVMSG::
                                        ERL$RELEASEMB::
                                        ERL$DEVICERR::
                                        ERL$DEVICTMO::
                                        ;       EXE$PWRTIMCHK::                 ; Put into QIO module              [02]
      05                                        RSB                             ; Return
                                        EXE$BUFFRQUOTA::
                                        EXE$BUFQUOPRC::
                                        SCH$WAKE::
50    01  D0                                   MOVL    #SS$_NORMAL, R0         ; Success code
      05                                        RSB
```

```
                     0007   108          .Sbttl  MUTEX Control Routines
                     0007   109
                     0007   110 ;++
                     0007   111 ; FUNCTIONAL DESCRIPTION:
                     0007   112 ;       Lock and unlock MUTEX database.  Since the Supervisor does not
                     0007   113 ; implement multi-programming, a 'MUTEX''-type lock is implemented via
                     0007   114 ; disabling AST delivery.
                     0007   115 ;
                     0007   116 ; CALLING SEQUENCE:
                     0007   117 ;       SCH$LOCKW -
                     0007   118 ;               JSB     SCH$LOCKW
                     0007   119 ;       SCH$UNLOCK -
                     0007   120 ;               JSB     SCH$UNLOCK
                     0007   121 ;
                     0007   122 ; INPUT PARAMETERS:
                     0007   123 ;       Ignored.  In Supervisor implmentation, which MUTEX is targeted for
                     0007   124 ; lock or unlock is irrelevant, since no data is accessed.
                     0007   125 ;
                     0007   126 ; IMPLICIT INPUTS:
                     0007   127 ;       none
                     0007   128 ;
                     0007   129 ; OUTPUT PARAMETERS:
                     0007   130 ;       none
                     0007   131 ;
                     0007   132 ; IMPLICIT OUTPUTS:
                     0007   133 ;       none
                     0007   134 ;
                     0007   135 ; COMPLETION CODES:
                     0007   136 ;       SS$_NORMAL
                     0007   137 ;
                     0007   138 ; SIDE EFFECTS:
                     0007   139 ;       SCH$LOCKW results in AST delivery for the current mode to be
                     0007   140 ;       disabled.  SCH$UNLOCK results in AST delivery being re-enabled.
                     0007   141 ;--
```

```
                                0007   143 ;
                                0007   144 ; Lock common data block for write
                                0007   145 ;
                                0007   146         .ENABL  LSB
                                0007   147
                                0007   148 SCH$LOCKW::
                     7E  50  7D 0007   149         MOVQ    R0,-(SP)                ; Save R0, R1
                         7E  D4 000A   150         CLRL    -(SP)                   ; Set to disable AST delivery
                         05  11 000C   151         BRB     10$                     ; Join common code
                                000E   152
                                000E   153 SCH$UNLOCK::
                     7E  50  7D 000E   154         MOVQ    R0,-(SP)                ; Save R0, R1
                         01  DD 0011   155         PUSHL   #1                      ; Set to enable AST delivery
                                0013   156
           00000000'9F   01  FB 0013   157 10$:    CALLS   #1, @#EXE$SETAST        ; Enable/disable AST delivery
                     50  8E  7D 001A   158         MOVQ    (SP)+,R0                ; Restore registers
                         05  001D   159         RSB                             ; Return with SETAST completion code
                                001E   160
                                001E   161         .DISABLE LSB
                                001E   162         .END
```

```
ERL$DEVICERR                    00000002 RG D   02
ERL$DEVICTMO                    00000002 RG D   02
ERL$RELEASEMB                   00000002 RG D   02
EXE$BUFFRQUOTA                  00000003 RG D   02
EXE$BUFQUOPRC                   00000003 RG D   02
EXE$SETAST                      ******** X      02
EXE$SNDEVMSG                    00000002 RG D   02
MT$CHECK_ACCESS                 00000000 RG D   02
SCH$LOCKW                       00000007 RG D   02
SCH$UNLOCK                      0000000E RG D   02
SCH$WAKE                        00000003 RG D   02
SS$_NORMAL                    = 00000001    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	0000001E (30.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ERL$DEVICERR	00000002-R	97 (1)	
ERL$DEVICTMO	00000002-R	98 (1)	
ERL$RELEASEMB	00000002-R	96 (1)	
EXE$BUFFRQUOTA	00000003-R	102 (1)	
EXE$BUFQUOPRC	00000003-R	103 (1)	
EXE$SETAST	00000000-XR		157 (1)
EXE$SNDEVMSG	00000002-R	95 (1)	
MT$CHECK_ACCESS	00000000-R	93 (1)	
SCH$LOCKW	00000007-R	148 (1)	
SCH$UNLOCK	0000000E-R	153 (1)	
SCH$WAKE	00000003-R	104 (1)	
SS$_NORMAL	=00000001		#-105 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	57 (1)	57 (1)
$SSDEF	21	57 (1)	57 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.11	00:00:00.34
Command processing	159	00:00:00.87	00:00:01.95
Pass 1	223	00:00:03.86	00:00:08.09
Symbol table sort	0	00:00:00.60	00:00:00.79
Pass 2	64	00:00:00.76	00:00:01.32
Symbol table output	2	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.01	00:00:00.02
Cross-reference output	7	00:00:00.08	00:00:00.08
Assembler run totals	492	00:00:06.33	00:00:12.62

The working set limit was 1000 pages.
19914 bytes (39 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 413 non-local and 1 local symbols.
162 source lines were read in Pass 1, producing 0 object records in Pass 2.
9 pages of virtual memory were used to define 7 macros.

Macro library statistics

Macro library name	Macros defined
DMA1:[SYS0.SYSMAINT]DS.MLB;218	0
DMA1:[SYS0.SYSMAINT]DIAG.MLB;953	0
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	4

464 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

```
MACRO/NOOBJECT/LIST=[DS.LIS]/CROSS/ENABLE=(DEBUG,TRACE) VMSDUMMY/UPDA=(VMSDUMMY.UPD,VMSDUMMY.ENH)+SYS$LIBRARY:LIB/LIBRARY+DMA1:[SYS0
```

Fiche	Frame	Sequence		
1	B1	1	Documentation	
1	B5	53	Map	
1	M8	103	VAX DIAGNOSTIC	
1	N8	104	*** ACPFDT ACP function dispatch	27-JUL-1984
1	D9	107	ACCESS AND CREATE ACP FUNCTION PROCESSIN	27-JUL-1984
1	E9	108	DEACCESS ACP FUNCTION PROCESSING	27-JUL-1984
1	F9	109	DELETE AND MODIFY ACP FUNCTION PROCESSIN	27-JUL-1984
1	G9	110	MOUNT ACP FUNCTION PROCESSING	27-JUL-1984
1	H9	111	READ AND WRITE BLOCK ACP FUNCTION PROCES	27-JUL-1984
1	K9	114	Symbol table	27-JUL-1984
1	M9	116	Psect synopsis	27-JUL-1984
1	N9	117	Cross reference	27-JUL-1984
1	D10	120	*** ANSI magtape RMS support	27-Jul-1984
1	G10	123	ansi$open	27-Jul-1984
1	I11	138	ansi$close	27-Jul-1984
1	K11	140	ansi$cachevbn	27-Jul-1984
1	B12	144	ansi$read	27-Jul-1984
1	E12	147	ansi$get	27-Jul-1984
1	M12	155	*** APT interface to APT control	27-JUL-1984
1	C13	158	DECLARATIONS	27-JUL-1984
1	D13	159	APT SUPPORT ROUTINES	27-JUL-1984
1	J13	167	Symbol table	27-JUL-1984
1	L13	167	Psect synopsis	27-JUL-1984
1	M13	168	Cross reference	27-JUL-1984
1	F14	174	*** ASSIGN assign device	27-JUL-1984
1	I14	177	ASSIGN I/O CHANNEL	27-JUL-1984
1	M14	181	TEST IF MAILBOX SPECIFIED	27-JUL-1984
1	N14	182	Symbol table	27-JUL-1984
1	C15	184	Psect synopsis	27-JUL-1984
1	D15	185	Cross reference	27-JUL-1984
1	G15	188	*** ASTCON AST control service	27-JUL-1984
1	I15	190	DECLARATIONS	27-JUL-1984
1	J15	191	EXE$SETAST - SET AST ENABLES	27-JUL-1984
1	K15	192	Symbol table	27-JUL-1984
1	L15	193	Psect synopsis	27-JUL-1984
1	M15	194	Cross reference	27-JUL-1984
1	C16	197	*** ASTDEL AST delivery	27-JUL-1984
1	E16	199	HISTORY ; DETAILED	27-JUL-1984
1	F16	200	Librarys and Equated Symbols	27-JUL-1984
1	G16	201	Data PSect Declarations	27-JUL-1984
1	H16	202	SCH$ASTDEL - AST DELIVERY INTERRUPT HAND	27-JUL-1984
1	D1	209	SCH$QAST - ENQUEUE AST CONTROL BLOCK FOR	27-JUL-1984
1	F1	211	Symbol table	27-JUL-1984
1	G1	212	Psect synopsis	27-JUL-1984
1	H1	213	Cross reference	27-JUL-1984
1	L1	217	*** ATTACH Handle ATTACH command	27-JUL-1984
2	C2	221	Libraries, Equated Symbols, Externals	27-JUL-1984
2	D2	222	Data Psect Declarations	27-JUL-1984
2	G2	225	Work Psect Declarations	27-JUL-1984
2	H2	226	DSV$ATTACH Attach command processor	27-JUL-1984
2	K2	229	DSX$ATTACH Process ATTACH command	27-JUL-1984
2	B3	233	Process invalid generic name error	27-JUL-1984
2	E3	236	Get name processing	27-JUL-1984
2	I3	240	Get device attributes	27-JUL-1984
2	K3	242	parse device name	27-JUL-1984
2	B4	246	PT_INTERPRET Decode PT-descriptor	27-JUL-1984
2	I4	253	DP$EXTRACT Make P-table printable	27-JUL-1984

Fiche	Frame	Sequence		
2	B5	259	GET_LINE Condition getline	27-JUL-1984
2	D5	261	FORMAT_PROMPT Format and load prompt	27-JUL-1984
2	F5	263	DSV$SHOWDEVICE Display device informatio	27-JUL-1984
2	H5	265	DSV$SHOWSELECT Display information selec	27-JUL-1984
2	J5	267	DSV$DEATTACH Deattach device	27-JUL-1984
2	K5	268	Deattach subtree	27-JUL-1984
2	L5	269	DSV$SELECT Select device for testing	27-JUL-1984
2	D6	274	DSV$DESELECT Deselect device for testing	27-JUL-1984
2	I6	279	SHOW_DEV Routine	27-JUL-1984
2	L6	282	INI$PTABLE Initialize P-table	27-JUL-1984
2	N6	284	Check ambiguous device	27-JUL-1984
2	C7	286	Locate /Adapter value	27-JUL-1984
2	E7	288	LOC$PTABLE Routine	27-JUL-1984
2	G7	290	LOC$PTDESC Locate PT-descriptor	27-JUL-1984
2	I7	292	INS$PTABLE Insert P-table	27-JUL-1984
2	K7	294	Symbol table	27-JUL-1984
2	B8	298	Psect synopsis	27-JUL-1984
2	C8	299	Cross reference	27-JUL-1984
2	N8	310	*** BOOTDRIVR non-interrupt I/O	27-JUL-1984
2	E9	314	Declarations	27-JUL-1984
2	G9	316	DRIVER FIXED DATA AREA	27-JUL-1984
2	H9	317	BOO$QIO - BOOTSTRAP QIO ROUTINE	27-JUL-1984
2	M9	322	BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QI	27-JUL-1984
2	B10	324	BOO$PURDPR - Purge UBA Buffered Datapath	27-JUL-1984
2	E10	327	BOO$SELECT - Select boot driver	27-JUL-1984
2	G10	329	Symbol table	27-JUL-1984
2	H10	330	Psect synopsis	27-JUL-1984
2	I10	331	Cross reference	27-JUL-1984
2	M10	335	- BOOTSTRAP FILEREAD IO MODULE	27-JUL-1984
2	E11	340	RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMB	27-JUL-1984
2	F11	341	BOO$CACHE_INIT - INIT FILEREAD CACHE	27-JUL-1984
2	I11	344	BOO$IMAGE_ATT - Get image attributes fro	27-JUL-1984
2	J11	345	SYS$ASSIGN, Dummy assign device system s	27-JUL-1984
2	K11	346	Symbol table	27-JUL-1984
2	L11	347	Cross reference	27-JUL-1984
2	N11	349	*** BUFCTL QIO buffer control	27-JUL-1984
2	D12	352	GET ONE BYTE OF DATA FROM USER BUFFER	27-JUL-1984
2	E12	353	PUT ONE BYTE OF DATA INTO USER'S BUFFER	27-JUL-1984
2	F12	354	INITIALIZE FOR SINGLE BYTE TRANSFERS	27-JUL-1984
2	G12	355	MOVE FROM USER BUFFER	27-JUL-1984
2	H12	356	MOVE TO USER BUFFER	27-JUL-1984
2	I12	357	FILL SYSTEM PTE WITH TRANSFER PTE	27-JUL-1984
2	J12	358	Symbol table	27-JUL-1984
2	K12	359	Psect synopsis	27-JUL-1984
2	L12	360	Cross reference	27-JUL-1984
2	N12	362	*** BUFFER GETBUF/RELBUF routines	27-JUL-1984
2	C13	364	DECLARATIONS	27-JUL-1984
2	D13	365	GET A VERTUAL MEMORY BUFFER.	27-JUL-1984
2	F13	367	RELEASE A MEMORY BUFFER.	27-JUL-1984
2	H13	369	Symbol table	27-JUL-1984
2	I13	370	Psect synopsis	27-JUL-1984
2	J13	371	Cross reference	27-JUL-1984
2	N13	375	*** CANCEL cancel QIO	27-JUL-1984
2	D14	378	CANCEL I/O ON CHANNEL	27-JUL-1984
2	G14	381	Symbol table	27-JUL-1984
2	I14	383	Psect synopsis	27-JUL-1984
2	J14	384	Cross reference	27-JUL-1984

Fiche	Frame	Sequence		
	M14	387	*** CHAN730 Channel services for NEBULA	27-JUL-1984
	C15	390	DECLARATIONS	27-JUL-1984
	H15	395	$CHANNEL CALL	27-JUL-1984
	J15	397	$CHANNEL CALL - ADAPTER INIT FUNCTION	27-JUL-1984
	K15	398	$CHANNEL CALL - BUS INIT FUNCTION	27-JUL-1984
	L15	399	$CHANNEL CALL - ABORT FUNCTION	27-JUL-1984
	M15	400	$CHANNEL CALL - ENABLE INTERRUPTS	27-JUL-1984
	N15	401	$CHANNEL CALL - PURGE	27-JUL-1984
	B16	402	$CHANNEL CALL - DISABLE INTERRUPTS	27-JUL-1984
	C16	403	$CHANNEL CALL - CLEAR ADAPTER	27-JUL-1984
	D16	404	$CHANNEL CALL - GET ADAPTER STATUS	27-JUL-1984
	E16	405	$CHANNEL CALL - SET DEFEAT PARITY	27-JUL-1984
	F16	406	$CHANNEL CALL - CLEAR DEFEAT PARITY	27-JUL-1984
	G16	407	$CHANNEL CALL - COMMON CALL RETURN	27-JUL-1984
	H16	408	$SETMAP CALL	27-JUL-1984
	D1	415	DSP$SHOWMBA	27-JUL-1984
	E1	416	DSP$SHOWUBI	27-JUL-1984
	F1	417	$SHOWCHAN CALL	27-JUL-1984
	G1	418	SHOW UBI STATUS	27-JUL-1984
	H1	419	SHOWBITS	27-JUL-1984
	I1	420	DETERMINE ADAPTER SPECIFIC STATUS	27-JUL-1984
	J1	421	DETERMINE GENERAL ADAPTER STATUS	27-JUL-1984
	K1	422	DETERMINE ADAPTER HARDWARE ERROR STATUS	27-JUL-1984
	M1	424	BUILD THE CHANNEL DATA BLOCK	27-JUL-1984
	N1	425	SET ADAPTER VECTOR(S)	27-JUL-1984
	B2	426	CLEAR ADAPTER VECTOR(S)	27-JUL-1984
	C2	427	INTERRUPT PRE-PROCESS	27-JUL-1984
	E2	429	Symbol table	27-JUL-1984
	G2	431	Psect synopsis	27-JUL-1984
	H2	432	Cross reference	27-JUL-1984
	C3	440	*** CHMK Change mode to kernel handler	27-JUL-1984
	E3	442	DECLARATIONS	27-JUL-1984
	F3	443	CHANGE MODE TO KERNEL ROUTINE	27-JUL-1984
	H3	445	ASTEXIT SYSTEM SERVICE	27-JUL-1984
	I3	446	Local intermediate dispatches	27-JUL-1984
	J3	447	Symbol table	27-JUL-1984
	K3	448	Psect synopsis	27-JUL-1984
	L3	449	Cross reference	27-JUL-1984
	C4	453	DIAGNOSTIC SUPERVISOR COMMAND LINE INTER	27-JUL-1984
	B5	465	Libraries, Macros, Equated Symbols	27-JUL-1984
	C5	466	Work Psect Storage	27-JUL-1984
	D5	467	Data Psect Storage	27-JUL-1984
	F5	469	Commands A-C	27-JUL-1984
	H5	471	Directory, Deattach, Deselect, Deposit C	27-JUL-1984
	J5	473	Commands Exit, Examine	27-JUL-1984
	K5	474	Help, InitPCS, Load, Next, Run	27-JUL-1984
	L5	475	Commands Set, Select, Show, Start, Summa	27-JUL-1984
	M5	476	End of Line Processing	27-JUL-1984
	C6	479	Illegal Command Processing	27-JUL-1984
	D6	480	Incomplete Command Processing	27-JUL-1984
	E6	481	Bad List Processing	27-JUL-1984
	F6	482	Bad Qualifier Processing	27-JUL-1984
	G6	483	Get Generic Device Name	27-JUL-1984
	H6	484	Deattach Processing	27-JUL-1984
	I6	485	Deselect and Select Processing	27-JUL-1984
	J6	486	Get a File Specification	27-JUL-1984
	L6	488	Start and Run Processing	27-JUL-1984

Fiche	Frame	Sequence		
	B7	491	Start and Run Qualifiers	27-JUL-1984
	D7	493	Show Parameters and Qualifiers	27-JUL-1984
	M7	502	Clear Parameters and Qualifiers	27-JUL-1984
	F8	508	Set Parameters and Qualifiers	27-JUL-1984
	C9	518	Examine Qualifiers and Parameters	27-JUL-1984
	I9	524	Deposit Parameters and Qualifiers	27-JUL-1984
	N9	529	Set Default Parameters	27-JUL-1984
	D10	532	DS$Cli Routine - Command Line Interprete	27-JUL-1984
	H10	536	CLI Action Routines.	27-JUL-1984
	L10	540	Null and Context-Saving Action Routines	27-JUL-1984
	N10	542	Success, Failure, Notnuf, Enuf Action Ro	27-JUL-1984
	B11	543	Commands A-C Action Routines	27-JUL-1984
	C11	544	Commands D-E Action Routines	27-JUL-1984
	E11	546	Commands H-R Action Routines	27-JUL-1984
	G11	548	Commands S-Z Action Routines	27-JUL-1984
	H11	549	CRD command	27-JUL-1984
	J11	551	Filespec, Optional, Required Action Rout	27-JUL-1984
	K11	552	Start and Run Qualifier Action Routines	27-JUL-1984
	M11	554	Event and Flags Action Routines	27-JUL-1984
	N11	555	Base, Address, Break Action Routines	27-JUL-1984
	B12	556	All, Default Action Routines	27-JUL-1984
	C12	557	Show Calls Routine ; [70]	27-JUL-1984
	D12	558	CRDTrace Action Routines ; [65]	27-JUL-1984
	E12	559	Bell, Binary, Halt, IEx Action Routines	27-JUL-1984
	I12	563	Loop, Oper, Prompt Quick Action Routines	27-JUL-1984
	J12	564	Search, Trace, Verify Action Routines	27-JUL-1984
	K12	565	Set/Show Page, Width Action Routine	27-JUL-1984
	L12	566	Set/Show QA Action Routines	27-JUL-1984
	M12	567	QADef Action Routine	27-JUL-1984
	N12	568	Efn, Next, Ascii Action Routines	27-JUL-1984
	B13	569	Register Action Routines	27-JUL-1984
	C13	570	Backup, Advance, Data Action Routines	27-JUL-1984
	D13	571	Long, Word, Byte, Hex, Dec, Oct Action R	27-JUL-1984
	E13	572	IF Action Routines	27-JUL-1984
	H13	575	Xdelta Action Routines	27-JUL-1984
	I13	576	Device, Brief, Adaptor Action Routines	27-JUL-1984
	J13	577	Show Select, Do_Cmd, Set Load, Show Load	27-JUL-1984
	K13	578	MM On, Off, and Show Action Routines	27-JUL-1984
	M13	580	Show Status, Show Support	27-JUL-1984
	B14	582	Symbol table	27-JUL-1984
	H14	588	Psect synopsis	27-JUL-1984
	I14	589	Cross reference	27-JUL-1984
4	B1	619	INTERVAL TIMER ROUTINES.	27-JUL-1984
4	F1	623	Macros and Equated Symbols	27-JUL-1984
4	J1	627	Data Psect Declarations	27-JUL-1984
4	K1	628	DS$CanWait - CANCEL DELAY ROUTINE	27-JUL-1984
4	L1	629	$WAKE - Wakeup from hibernate	27-JUL-1984
4	M1	630	DSX$WaitMS - MILLISECOND DELAY ROUTINE	27-JUL-1984
4	B2	632	DSX$WaitUC - MICROSECOND DELAY ROUTINE	27-JUL-1984
4	D2	634	EXE$HIBER - HIBERNATE SYSTEM SERVICE ROU	27-JUL-1984
4	F2	636	EXE$CANTIM - CANCEL TIMER SYSTEM SERVICE	27-JUL-1984
4	H2	638	DSR$SETIMR_INI - SET TIMER QUEUE INITIAL	27-JUL-1984
4	J2	640	EXE$Setimr - SET TIMER SYSTEM SERVICE.	27-JUL-1984
4	L2	642	DSI$TimSrv - INTERVAL TIMER INTERRUPT SE	27-JUL-1984
4	N2	644	DSI$TIMER - Level 7 timer interrupt serv	27-JUL-1984
4	E3	648	CLK$RE_INIT - SYSTEM TIMER INITIALIZATIO	27-JUL-1984
4	G3	650	Symbol table	27-JUL-1984

Fiche	Frame	Sequence		
4	K3	654	Psect synopsis	27-JUL-1984
4	L3	655	Cross reference	27-JUL-1984
4	F4	662	*** COMDRVSUB driver support routines	27-JUL-1984
4	I4	665	COM$DELATTNAST - DELIVER ATTENTION ASTS	27-JUL-1984
4	J4	666	COM$FLUSHATTNS - FLUSH ATTENTION AST LIS	27-JUL-1984
4	K4	667	COM$POST - POST I.O COMPLETION INDEPENDE	27-JUL-1984
4	L4	668	COM$DRVDEALMEM - DEALLOCATE DRIVER MEMOR	27-JUL-1984
4	M4	669	COM$SETATTNAST - SET UP ATTENTION AST	27-JUL-1984
4	B5	671	Symbol table	27-JUL-1984
4	D5	673	Psect synopsis	27-JUL-1984
4	E5	674	Cross reference	27-JUL-1984
4	H5	677	*** CONFIG Configure load device	27-JUL-1984
4	B6	684	INI$LOAD_DEVICE Make ptables for consol	27-JUL-1984
4	E6	687	DSP$CONFIG_LOAD Configure load device	27-JUL-1984
4	L6	694	Common routines	27-JUL-1984
4	B7	697	configure RB730	27-JUL-1984
4	E7	700	Attach UDA-50/LESI load path	27-JUL-1984
4	H7	703	Symbol table	27-JUL-1984
4	K7	706	Psect synopsis	27-JUL-1984
4	L7	707	Cross reference	27-JUL-1984
4	G8	715	*** CONSOLE Console I/O handler	27-JUL-1984
4	L8	720	Libraries, Equated Symbols, Macros	27-JUL-1984
4	M8	721	Work Psect Declarations ; [13]	27-JUL-1984
4	N8	722	Data Psect Declarations ; [13]	27-JUL-1984
4	E9	726	VRSetWidth Routine - Set terminal width	27-JUL-1984
4	G9	728	VRShowWidth Routine - Show terminal widt	27-JUL-1984
4	I9	730	KB_Poll Routine - Keyboard ready bit che	27-JUL-1984
4	K9	732	RInput Routine - Flag driven terminal in	27-JUL-1984
4	N9	735	RType Routine - Retype input line ('^R'	27-JUL-1984
4	C10	737	Out Routines - Flag driven terminal outp	27-JUL-1984
4	G10	741	WriteVBlk Routine - Console write driver	27-JUL-1984
4	I10	743	ReadVBlk Routine - Console read driver	27-JUL-1984
4	M10	747	SIZE_TERM Routine to size the console te	27-JUL-1984
4	C11	750	Symbol table	27-JUL-1984
4	F11	753	Psect synopsis	27-JUL-1984
4	G11	754	Cross reference	27-JUL-1984
4	B12	762	*** Loading and Unloading the CRD Image	27-Jul-1984
4	D12	764	Libraries	27-Jul-1984
4	E12	765	Other Literals	27-Jul-1984
4	F12	766	External Routines	27-Jul-1984
4	G12	767	Psect Definitions	27-Jul-1984
4	H12	768	Local ASCIC Bindings	27-Jul-1984
4	I12	769	Own Storage	27-Jul-1984
4	K12	771	DSR$Load_CRD Routine	27-Jul-1984
4	I13	782	DSR$Unload_CRD Routine	27-Jul-1984
4	N13	787	DSR$Restore_CRD_Vectors Routine	27-Jul-1984
4	C14	789	Exchange_Dispatch_Vectors Routine	27-Jul-1984
4	G14	793	CRD_Exit Routine	27-Jul-1984
4	I14	795	CRD_Error Routine	27-Jul-1984
4	N14	800	Internal_Error Routine	27-Jul-1984
4	H15	807	DSR$Print_TypeCode_Name Routine	27-Jul-1984
4	N15	813	*** CVRTIM time conversion routines	27-JUL-1984
4	F16	818	CONVERT BINARY TIME TO ASCII STRING	27-JUL-1984
4	H16	820	CONVERT ASCII STRING TO BINARY TIME	27-JUL-1984
5	B1	825	CONVERT BINARY TIME TO NUMERIC TIME	27-JUL-1984
5	F1	829	Symbol table	27-JUL-1984
5	G1	830	Cross reference	27-JUL-1984

Fiche	Frame	Sequence		
5	J1	833	- Converts ASCII to RAD50	27-JUL-1984
5	L1	835	DECLARATIONS	27-JUL-1984
5	M1	836	CVTFILNAM - CONVERT FILE NAME FROM ASCII	27-JUL-1984
5	C2	839	STORER50BYTE - CONVERT AND STORE ASCII B	27-JUL-1984
5	D2	840	PACKRAD50 - PACK 3 BYTES OF RAD50 CHARAC	27-JUL-1984
5	E2	841	ASCIITORAD50 - CONVERT ASCII CHARACTER T	27-JUL-1984
5	F2	842	Symbol table	27-JUL-1984
5	G2	843	Cross reference	27-JUL-1984
5	I2	845	*** DASSGN deassign QIO channel	27-JUL-1984
5	L2	848	DEASSIGN I/O CHANNEL	27-JUL-1984
5	N2	850	Symbol table	27-JUL-1984
5	C3	852	Psect synopsis	27-JUL-1984
5	D3	853	Cross reference	27-JUL-1984
5	G3	856	- CONSOLE TU58 BOOT DRIVER	27-JUL-1984
5	J3	859	DECLARATIONS	27-JUL-1984
5	L3	861	DD_SELECT - Select correct CPUs for this	27-JUL-1984
5	M3	862	Console TU58 Driver	27-JUL-1984
5	E4	867	Character transmission	27-JUL-1984
5	I4	871	Symbol table	27-JUL-1984
5	K4	873	Cross reference	27-JUL-1984
5	C5	878	EXAMINE, DEPOSIT, AND BREAKPOINT SUBROUT	27-JUL-1984
5	F5	881	DECLARATIONS	27-JUL-1984
5	M5	888	SET BASE ADDRESS SUBROUTINE	27-JUL-1984
5	N5	889	SHOW BASE ADDRESS SUBROUTINE ; [13]	27-JUL-1984
5	B6	890	SET DEFAULTS SUBROUTINE	27-JUL-1984
5	D6	892	SHOW DEFAULTS SUBROUTINE	27-JUL-1984
5	F6	894	EXAMINE MEMORY AND REGISTERS SUBROUTINE	27-JUL-1984
5	J6	898	VRTYPEXAMINE Type out location EXAMINE'd	27-JUL-1984
5	N6	902	DEPOSIT MEMORY AND REGISTERS SUBROUTINE	27-JUL-1984
5	F7	906	SET BREAKPOINT SUBROUTINE	27-JUL-1984
5	G7	908	CLEAR BREAKPOINT SUBROUTINE	27-JUL-1984
5	I7	910	DSP$REMOVEBREAK Remove breakpoints	27-JUL-1984
5	J7	911	SHOW BREAKPOINTS SUBROUTINE	27-JUL-1984
5	K7	912	FIND_BPT FIND BPT IN BPT TABLE	27-JUL-1984
5	L7	913	DEBUGGER CONDITION HANDLER	27-JUL-1984
5	M7	914	BREAKPOINT CONDITION SUBROUTINE	27-JUL-1984
5	C8	917	TBIT CONDITION SUBROUTINE	27-JUL-1984
5	E8	919	TEMPORARY BREAKPOINT REMOVAL SUBROUTINE	27-JUL-1984
5	F8	920	BREAKPOINT REPLACEMEMT SUBROUTINE	27-JUL-1984
5	G8	921	FETCH_STORE COPY BYTE ROUTINE	27-JUL-1984
5	M8	927	Handle EXAMINE/DEPOSIT of IPRs in KERNEL	27-JUL-1984
5	N8	928	FETCH_STOR_PREG Move data to/from IPR	27-JUL-1984
5	B9	929	DSV$DEBUG - Process DEBUG Command [14	27-JUL-1984
5	D9	931	Symbol table	27-JUL-1984
5	H9	935	Cross reference	27-JUL-1984
5	E10	945	*** DEVALC device allocation routines	27-JUL-1984
5	H10	948	ALLOCATE DEVICE	27-JUL-1984
5	K10	951	DEALLOCATE DEVICE	27-JUL-1984
5	M10	953	LOCK I/O DATA BASE AND SEARCH FOR DEVICE	27-JUL-1984
5	N10	954	DECREMENT REFERENCE COUNT, CLEAR OWNERSH	27-JUL-1984
5	B11	955	DSX$MOUNT - Mount a device	27-JUL-1984
5	C11	956	DSX$DISMOUNT - Dismount a device	27-JUL-1984
5	D11	957	Symbol table	27-JUL-1984
5	G11	960	Cross reference	27-JUL-1984
5	K11	964	*** DEVICE Handle PTABLEs	27-JUL-1984
5	N11	967	DS$GPHARD Retreive address of P-table	27-JUL-1984
5	B12	968	DSP$GEN_PTABLES Setup UUT's to test	27-JUL-1984

Fiche	Frame	Sequence		
5	D12	970	Symbol table	27-JUL-1984
5	E12	971	Psect synopsis	27-JUL-1984
5	F12	972	Cross reference	27-JUL-1984
5	J12	976	*** DIRECTORY Handle DIRECTORY command	27-Jul-1984
5	N12	980	Put filename to buffer, if matches pattern	27-Jul-1984
5	D13	983	RAD50	27-Jul-1984
5	F13	985	format_ods1	27-Jul-1984
5	H13	987	format_ods2	27-Jul-1984
5	K13	990	RT-11 directory	27-Jul-1984
5	B14	994	ANSI directory	27-Jul-1984
5	G14	999	ods directory	27-Jul-1984
5	M14	1005	Main code	27-Jul-1984
5	G15	1012	Master routine	27-Jul-1984
5	I15	1014	*** DISPAT Diagnostic test sequence cont	27-JUL-1984
5	N15	1019	Libraries and Symbol Definitions	27-JUL-1984
5	B16	1020	Work Declarations	27-JUL-1984
5	C16	1021	Data Declarations	27-JUL-1984
5	F16	1024	Dispatch Routine	27-JUL-1984
6	D1	1033	DSX$EndPass Routine	27-JUL-1984
6	G1	1036	DS_Cleanup Routine	27-JUL-1984
6	J1	1039	DSX$DoSummary and VRSummary Routines	27-JUL-1984
6	M1	1042	VRShowStatus Routine	27-JUL-1984
6	B2	1044	Symbol table	27-JUL-1984
6	F2	1048	Psect synopsis	27-JUL-1984
6	G2	1049	Cross reference	27-JUL-1984
6	D3	1059	- RL01/2 BOOT DRIVER	27-JUL-1984
6	F3	1061	DECLARATIONS	27-JUL-1984
6	H3	1063	RL01/2 Bootstrap driver code	27-JUL-1984
6	L3	1067	Symbol table	27-JUL-1984
6	M3	1068	Cross reference	27-JUL-1984
6	B4	1070	- RK06/7 BOOT DRIVER	27-JUL-1984
6	D4	1072	DECLARATIONS	27-JUL-1984
6	G4	1075	RK06/7 Bootstrap driver code	27-JUL-1984
6	J4	1078	ECC - PERFORM ECC ERROR CORRECTION	27-JUL-1984
6	L4	1080	Symbol table	27-JUL-1984
6	M4	1081	Cross reference	27-JUL-1984
6	B5	1083	- RB730:RB02/RB80 BOOT DRIVER	27-JUL-1984
6	E5	1086	DECLARATIONS	27-JUL-1984
6	H5	1089	RB730:RB02/RB80 Bootstrap driver code	27-JUL-1984
6	M5	1094	Symbol table	27-JUL-1984
6	N5	1095	Cross reference	27-JUL-1984
6	D6	1098	- DR32 interrupt handler	27-JUL-1984
6	F6	1100	DECLARATIONS	27-JUL-1984
6	H6	1102	DR$INT - DR32 INTERRUPT HANDLER	27-JUL-1984
6	J6	1104	Symbol table	27-JUL-1984
6	K6	1105	Cross reference	27-JUL-1984
6	M6	1107	*** DSLOAD DS$LOAD routine	27-Jul-1984
6	C7	1110	DSX$LOAD routine	27-Jul-1984
6	K7	1118	*** DSVECS Module	27-JUL-1984
6	B8	1122	Libraries and Macros	27-JUL-1984
6	C8	1123	The Dispatch Vectors	27-JUL-1984
6	E8	1125	Symbol table	27-JUL-1984
6	F8	1126	Cross reference	27-JUL-1984
6	H8	1128	*** ENTRY DS service entry vectors	27-JUL-1984
6	N8	1134	Libraries and Equated Symbols	27-JUL-1984
6	B9	1135	Data Psect Declarations	27-JUL-1984
6	C9	1136	DS Entry Points	27-JUL-1984

Fiche	Frame	Sequence		
6	B12	1174	DS$SRVDISP_TABLE - Dispatch table for VD	27-JUL-1984
6	C12	1175	DSX$SERVICE_DISPATCHER - Service Dispatc	27-JUL-1984
6	D12	1176	Resrvd_Entry Routine - For entry errors	27-JUL-1984
6	F12	1178	Symbol table	27-JUL-1984
6	I12	1181	Psect synopsis	27-JUL-1984
6	J12	1182	Cross reference	27-JUL-1984
6	E13	1190	*** ERROR Error routines	27-JUL-1984
6	I13	1194	Libraries, Macros, Equated Symbols	27-JUL-1984
6	J13	1195	Work Psect Declarations	27-JUL-1984
6	K13	1196	Data Psect Declarations	27-JUL-1984
6	F14	1204	DSX$ErrSoft Routine	27-JUL-1984
6	H14	1206	DSX$ErrHard Routine	27-JUL-1984
6	I14	1207	DSX$ErrPrep Routine [17]	27-JUL-1984
6	K14	1209	DSX$ErrDev Routine	27-JUL-1984
6	M14	1211	DSX$ErrSys Routine	27-JUL-1984
6	B15	1213	RRtpError Routine	27-JUL-1984
6	F15	1217	DS_TestID Routine	27-JUL-1984
6	G15	1218	DS_ErrSup Routine	27-JUL-1984
6	J15	1221	DSR$Completion Routine	27-JUL-1984
6	L15	1223	Symbol table	27-JUL-1984
6	C16	1227	Cross reference	27-JUL-1984
6	K16	1235	*** EVENT flag system services	27-Jul-1984
6	L16	1236	Declarations	27-Jul-1984
7	B1	1237	check for legal event flag number	27-Jul-1984
7	C1	1238	set event flag routine	27-Jul-1984
7	E1	1240	Clear event flag routine	27-Jul-1984
7	G1	1242	post event flag	27-Jul-1984
7	I1	1244	read event flags routine	27-Jul-1984
7	K1	1246	wait for single event flag routine	27-Jul-1984
7	M1	1248	wait for logical or of event flags routine	27-Jul-1984
7	B2	1250	wait for logical and of event flags routine	27-Jul-1984
7	E2	1253	*** EXCEPT Machine Exception handling	27-JUL-1984
7	I2	1257	Libraries, External & Equated Symbols, M	27-JUL-1984
7	J2	1258	Work Psect Declarations	27-JUL-1984
7	K2	1259	Data Psect Declarations	27-JUL-1984
7	F3	1267	COND_HANDLR ROUTINE - HANDLE UNWANTED EX	27-JUL-1984
7	I3	1270	DSX$PrintSig Routine - Format Signal Arr	27-JUL-1984
7	D4	1278	FIND_USERPC Find USER PC on stack	27-JUL-1984
7	F4	1280	TST$MChk Routine - Machine Check Handler	27-JUL-1984
7	H4	1282	HARDWARE EXCEPTION ENTRY POINTS	27-JUL-1984
7	B5	1289	SEARCH FOR CONDITION HANDLER	27-JUL-1984
7	E5	1292	DSX$SETPRIEXV - ESTABLISH PRIMARY EXCEPT	27-JUL-1984
7	F5	1293	Symbol table	27-JUL-1984
7	K5	1298	Psect synopsis	27-JUL-1984
7	L5	1299	Cross reference	27-JUL-1984
7	J6	1310	*** - FORMATTED ASCII OUTPUT	27-JUL-1984
7	M6	1313	Libraries, Equated Symbols	27-JUL-1984
7	C7	1316	FAO - Main program	27-JUL-1984
7	H7	1321	GETCHAR - Routine to get next char from	27-JUL-1984
7	J7	1323	GETCOUNT - Routine to get repeat-count o	27-JUL-1984
7	L7	1325	CVTASC - Insert ASCII string	27-JUL-1984
7	B8	1328	CVTNUM - Convert numeric parameter to AS	27-JUL-1984
7	F8	1332	QUICKSERVE - Small service routines	27-JUL-1984
7	I8	1335	PERCENT - Time directives and plural 'S'	27-JUL-1984
7	L8	1338	HANDLER - Condition handler	27-JUL-1984
7	N8	1340	Symbol table	27-JUL-1984
7	B9	1341	Psect synopsis	27-JUL-1984

Fiche	Frame	Sequence		
7	C9	1342	Cross reference	27-JUL-1984
7	F9	1345	- FILES11 LEVEL 1 & 2 FILE READING ROUTI	27-JUL-1984
7	I9	1348	DECLARATIONS	27-JUL-1984
7	L9	1351	FIL$OPENFILE - RETURN FILE HEADER AND ST	27-JUL-1984
7	[illegible]10	1356	FIL$CACHE_INIT - INIT FILEREAD CACHE	27-JUL-1984
7	F10	1358	FIL$CACHE_TRUNC - TRUNCATE FILEREAD CACH	27-JUL-1984
7	G10	1359	ASSIGN_DEV - ASSIGN A DEVICE AND RETURN	27-JUL-1984
7	H10	1360	STORE3DIGITS - STORE 3 ASCII DIGITS	27-JUL-1984
7	I10	1361	FORMDIRSTRING - GET A DIRECTORY STRING	27-JUL-1984
7	K10	1363	MOUNT - MOUNT THE VOLUME, INIT FOR FILE	27-JUL-1984
7	N10	1366	FINDFILID - FIND FILE ID FOR SPECIFIED F	27-JUL-1984
7	G11	1372	FIL$FINDFILID - STRUCTURE LEVEL 2	27-JUL-1984
7	J11	1375	FIL$FINDFILID - STRUCTURE LEVEL 1	27-JUL-1984
7	L11	1377	READ_DIR_LBN - READ NEXT DIRECTORY LBN	27-JUL-1984
7	M11	1378	RDCHKFILHDR - READ AND CHECK FILE HEADER	27-JUL-1984
7	C12	1381	READVBN, WRITEVBN - READ/WRITE VIRTUAL B	27-JUL-1984
7	E12	1383	INIRTRVPTRSCAN - INITIALIZE RETRIEVAL PO	27-JUL-1984
7	F12	1384	GETRTRVPTR - CONVERT NEXT RETRIEVAL POIN	27-JUL-1984
7	H12	1386	STATBLK - GET FILE STATISTICS BLOCK	27-JUL-1984
7	K12	1389	FIL$CHKFILHDR - CHECK FILE HEADER VALIDI	27-JUL-1984
7	M12	1391	CHECKSUM - VALIDATE A CHECKSUM	27-JUL-1984
7	N12	1392	Symbol table	27-JUL-1984
7	C13	1394	Psect synopsis	27-JUL-1984
7	D13	1395	Cross reference	27-JUL-1984
7	J13	1401	*** FLAGS Set/Clear/Show flags	27-JUL-1984
7	N13	1405	Libraries, Macros, Equated Symbols	27-JUL-1984
7	B14	1406	Data Psect Declarations	27-JUL-1984
7	C14	1407	VRSetFlg Routine - Set control flags	27-JUL-1984
7	E14	1409	VRClrFlg Routine - Clear control flags	27-JUL-1984
7	G14	1411	VRSetEF Routine - Set event flags	27-JUL-1984
7	I14	1413	VRClrEF Routine - Clear event flags	27-JUL-1984
7	K14	1415	VRShowFlg Routine - Show control flags	27-JUL-1984
7	M14	1417	VRShowEF Routine - Show event flags	27-JUL-1984
7	B15	1419	Symbol table	27-JUL-1984
7	D15	1421	Cross reference	27-JUL-1984
7	I15	1426	*** FRKCTL QIO fork control	27-JUL-1984
7	K15	1428	CREATE I/O DRIVER FORK PROCESS	27-JUL-1984
7	L15	1429	CREATE FORK PROCESS	27-JUL-1984
7	M15	1430	SOFTWARE INTERRUPT FORK DISPATCHER	27-JUL-1984
7	B16	1432	Symbol table	27-JUL-1984
7	C16	1433	Psect synopsis	27-JUL-1984
7	D16	1434	Cross reference	27-JUL-1984
7	F16	1436	- SYSTEM SERVICE GET TIME	27-JUL-1984
7	H16	1438	GET TIME	27-JUL-1984
7	I16	1439	Symbol table	27-JUL-1984
7	J16	1440	Cross reference	27-JUL-1984
7	L16	1442	*** GTCHAN Get channel information	27-JUL-1984
8	C1	1444	GET I/O CHANNEL DEVICE INFORMATION	27-JUL-1984
8	E1	1446	Symbol table	27-JUL-1984
8	G1	1448	Cross reference	27-JUL-1984
8	I1	1450	*** HELP Handle HELP command	27-Jul-1984
8	D2	1458	Help control routine	27-Jul-1984
8	L2	1466	do help stuff	27-Jul-1984
8	H3	1475	check for key on line	27-Jul-1984
8	L3	1479	print key list	27-Jul-1984
8	N3	1481	print help text	27-Jul-1984
8	D4	1484	List other keys	27-Jul-1984

Fiche	Frame	Sequence		
8	F4	1486	PrintHeader	27-Jul-1984
8	H4	1488	print_otherhelp main code	27-Jul-1984
8	L4	1492	Read random record	27-Jul-1984
8	B5	1495	compare keys	27-Jul-1984
8	G5	1500	skip_blanks	27-Jul-1984
8	I5	1502	scan_word	27-Jul-1984
8	L5	1505	find_character	27-Jul-1984
8	N5	1507	print blank line	27-Jul-1984
8	C6	1509	- Interpreted code for TSP	27-JUL-1984
8	F6	1512	DECLARATIONS	27-JUL-1984
8	H6	1514	Device description database	27-JUL-1984
8	I10	1567	Symbol table	27-JUL-1984
8	N10	1572	Psect synopsis	27-JUL-1984
8	B11	1573	Cross reference	27-JUL-1984
8	M11	1584	*** IOBASE I/O data base	27-JUL-1984
8	D12	1588	DECLARATIONS	27-JUL-1984
8	H12	1592	Device description database	27-JUL-1984
8	N12	1598	Symbol table	27-JUL-1984
8	F13	1603	Psect synopsis	27-JUL-1984
8	G13	1604	Cross reference	27-JUL-1984
8	H15	1631	*** IOPOST I/O completion posting	27-JUL-1984
8	J15	1633	HISTORY ; DETAILED	27-JUL-1984
8	K15	1634	DECLARATIONS	27-JUL-1984
8	L15	1635	I/O COMPLETION POSTING	27-JUL-1984
8	B16	1638	VIRTUAL I/O COMPLETION	27-JUL-1984
8	D16	1640	QUEUE NEXT SEGMENT	27-JUL-1984
8	G16	1643	BUFFERED READ COMPLETION AST ROUTINE	27-JUL-1984
8	I16	1645	DIRECT I/O COMPLETION AST ROUTINE	27-JUL-1984
8	K16	1647	MOVE DATA TO USER BUFFER	27-JUL-1984
8	L16	1648	UNLOCK AREAS IN IRPE'S	27-JUL-1984
9	C1	1650	Symbol table	27-JUL-1984
9	G1	1654	Psect synopsis	27-JUL-1984
9	H1	1655	Cross reference	27-JUL-1984
9	M1	1660	*** IOSNPG services for QIO	27-JUL-1984
9	D2	1664	CANCEL I/O ON CHANNEL	27-JUL-1984
9	E2	1665	FILL DIAGNOSTIC BUFFER	27-JUL-1984
9	F2	1666	RELEASE I/O CHANNEL	27-JUL-1984
9	G2	1667	REQUEST I/O CHANNEL	27-JUL-1984
9	H2	1668	I/O REQUEST COMPLETION PROCESSING	27-JUL-1984
9	I2	1669	INITIATE I/O FUNCTION ON DEVICE	27-JUL-1984
9	J2	1670	RELEASE BUFFERED DATAPATH	27-JUL-1984
9	K2	1671	REQUEST BUFFERED DATAPATH	27-JUL-1984
9	L2	1672	RELEASE UNIBUS MAP REGISTERS	27-JUL-1984
9	M2	1673	REQUEST UNIBUS MAP REGISTERS	27-JUL-1984
9	N2	1674	ALTER UBA MAP REGISTER BITMAP	27-JUL-1984
9	B3	1675	SEARCH MAP REGISTER BITMAP AND ALLOCATE	27-JUL-1984
9	C3	1676	RETURN TO CALLER	27-JUL-1984
9	D3	1677	WAITFOR INTERRUPT OR TIMEOUT AND KEEP CH	27-JUL-1984
9	E3	1678	WAITFOR INTERRUPT OR TIMEOUT AND RELEASE	27-JUL-1984
9	F3	1679	ALLOCATE SYSTEM PAGE TABLE	27-JUL-1984
9	G3	1680	Symbol table	27-JUL-1984
9	I3	1682	Psect synopsis	27-JUL-1984
9	J3	1683	Cross reference	27-JUL-1984
9	N3	1687	*** IOSPGD services for QIO	27-JUL-1984
9	D4	1690	CONVERT DEVICE NAME AND UNIT	27-JUL-1984
9	E4	1691	FIND FREE I/O CHANNEL	27-JUL-1984
9	F4	1692	SEARCH FOR DEVICE	27-JUL-1984

Fiche	Frame	Sequence		
9	I4	1695	UNLOCK I/O DATA BASE AND RETURN STATUS	27-JUL-1984
9	J4	1696	VERIFY I/O CHANNEL NUMBER	27-JUL-1984
9	K4	1697	Symbol table	27-JUL-1984
9	N4	1700	Cross reference	27-JUL-1984
9	D5	1703	*** IOSRAM random access I/O service rou	27-JUL-1984
9	G5	1706	APPLY ECC CORRECTION	27-JUL-1984
9	I5	1708	CONVERT LOGICAL BLOCK TO PHYSICAL ADDRES	27-JUL-1984
9	J5	1709	MAP VIRTUAL TO LOGICAL BLOCK	27-JUL-1984
9	M5	1712	UPDATE TRANSFER PARAMETERS	27-JUL-1984
9	N5	1713	SENSE DISK'S SIZE FDT ROUTINE	27-JUL-1984
9	B6	1714	Symbol table	27-JUL-1984
9	D6	1716	Cross reference	27-JUL-1984
9	G6	1719	*** KERNEL Main control routine	27-JUL-1984
9	I7	1734	Libraries and Macros	27-JUL-1984
9	J7	1735	Equated Symbols	27-JUL-1984
9	L7	1737	Data Psect Declarations	27-JUL-1984
9	M7	1738	Work Psect Declarations	27-JUL-1984
9	C8	1741	Data Psect Declarations	27-JUL-1984
9	G8	1745	SHELL OF STAND ALONE SUPERVISOR MEMORY.	27-JUL-1984
9	I8	1747	DIAGNOSTIC SUPERVISOR BOOTSTRAP ROUTINE.	27-JUL-1984
9	N8	1752	USEP KERNEL ROUTINE	27-JUL-1984
9	H9	1759	COMMON KERNEL REFRESH ENTRY POINT.	27-JUL-1984
9	M9	1764	INIT_CONTEXT Initialize process context	27-JUL-1984
9	D10	1768	CONTROL-C AST HANDLER	27-JUL-1984
9	E10	1769	KEYBOARD CHECK ROUTINE	27-JUL-1984
9	I10	1773	DSX$CNTRLC Program enable for Control-C	27-JUL-1984
9	J10	1774	DSV$EXIT, Process EXIT command	27-JUL-1984
9	L10	1776	EXIT$HANDLR Process EXIT handler	27-JUL-1984
9	N10	1778	DSR$Check_MenuTest_Off_Set Routine	27-JUL-1984
9	C11	1780	DSR$Check_AutoTest_Off_Set Routine	27-JUL-1984
9	G11	1784	DSX$GETTERM Get User Terminal Character	27-JUL-1984
9	H11	1785	Symbol table	27-JUL-1984
9	B12	1792	Psect synopsis	27-JUL-1984
9	C12	1793	Cross reference	27-JUL-1984
9	D13	1807	*** LOAD Set/Show load	27-JUL-1984
9	G13	1810	Libraries, Equated Symbols	27-JUL-1984
9	H13	1811	Work Psect Declarations	27-JUL-1984
9	I13	1812	Data Psect Declarations	27-JUL-1984
9	J13	1813	DSV$SETLOAD SET THE DEFAULT LOAD DEVICE/	27-JUL-1984
9	L13	1815	DSV$SHOWLOAD Display default load device	27-JUL-1984
9	M13	1816	DSR$IFLOADDEV Is it the load device?	27-JUL-1984
9	N13	1817	DSV$LOAD PROGRAM IMAGE LOAD SUBROUTINE	27-JUL-1984
9	D14	1820	Symbol table	27-JUL-1984
9	F14	1822	Psect synopsis	27-JUL-1984
9	G14	1823	Cross reference	27-JUL-1984
9	M14	1829	*** LODMAP load adapter map registers	27-JUL-1984
9	D15	1833	LOAD UNIBUS ADAPTER MAP REGISTERS	27-JUL-1984
9	F15	1835	GET PFN FROM INVALID PTE	27-JUL-1984
9	G15	1836	Symbol table	27-JUL-1984
9	H15	1837	Psect synopsis	27-JUL-1984
9	I15	1838	Cross reference	27-JUL-1984
9	K15	1840	*** LOOP Loop control	27-JUL-1984
9	B16	1844	Declarations	27-JUL-1984
9	D16	1846	DSX$Escape Routine	27-JUL-1984
9	F16	1848	DSX$BgnSub Routine	27-JUL-1984
9	J16	1852	DSX$EndSub Routine	27-JUL-1984
10	E1	1858	DSX$CkLoop Routine	27-JUL-1984

Fiche	Frame	Sequence		
10	G1	1860	DSX$InLoop Routine	27-JUL-1984
10	I1	1862	Symbol table	27-JUL-1984
10	L1	1865	Psect synopsis	27-JUL-1984
10	M1	1866	Cross reference	27-JUL-1984
10	H2	1874	- MASSBUS ADAPTER INTERRUPT DISPATCHER	27-JUL-1984
10	K2	1877	MASSBUS ADAPTER INTERRUPT DISPATCHER	27-JUL-1984
10	M2	1879	MASSBUS ADAPTER INITIALIZATION	27-JUL-1984
10	N2	1880	Symbol table	27-JUL-1984
10	C3	1882	Cross reference	27-JUL-1984
10	E3	1884	*** MCHK730 machine check formatter	27-Jul-1984
10	G3	1886	CHK$MCHK	27-Jul-1984
10	L3	1891	DSR$FAULT_CLEAR	27-Jul-1984
10	B4	1894	*** MEMALC dynamic memory allocation	27-JUL-1984
10	G4	1899	ALLOCATE MEMORY AND CONDITIONALLY WAIT	27-JUL-1984
10	J4	1902	ALLOCATE NONPAGED DYNAMIC MEMORY	27-JUL-1984
10	L4	1904	GENERAL ALLOCATE MEMORY SUBROUTINE	27-JUL-1984
10	M4	1905	DEALLOCATE NONPAGED DYNAMIC MEMORY	27-JUL-1984
10	N4	1906	CHECK BLOCK PARAMETERS SUBROUTINE	27-JUL-1984
10	B5	1907	GENERAL DEALLOCATION SUBROUTINE	27-JUL-1984
10	C5	1908	MEMPOOL$INIT ;USER MODE MEMORY INITIAL	27-JUL-1984
10	E5	1910	Symbol table	27-JUL-1984
10	G5	1912	Cross reference	27-JUL-1984
10	K5	1916	*** MEMMGT Memory management setup/contr	27-JUL-1984
10	B6	1920	Libraries, Macros, and External Symbols	27-JUL-1984
10	C6	1921	Own Storage	27-JUL-1984
10	D6	1922	Data Storage	27-JUL-1984
10	E6	1923	MapMem Routine - Map all of memory	27-JUL-1984
10	K6	1929	MAPMEM$IOSPACE Setup page tables for I/O	27-JUL-1984
10	M6	1931	ACCESSABLE check a page for accessabilit	27-JUL-1984
10	N6	1932	STACKPROT	27-JUL-1984
10	B7	1933	MAP FREE MEMORY PROCEDURE.	27-JUL-1984
10	D7	1935	GET A BLOCK OF VIRTUAL MEMORY.	27-JUL-1984
10	G7	1938	RELEASE A BLOCK OF VIRTUAL MEMORY.	27-JUL-1984
10	J7	1941	TURN MEMORY MANAGEMENT ON.	27-JUL-1984
10	K7	1942	TURN MEMORY MANAGEMENT OFF.	27-JUL-1984
10	L7	1943	DSR$MMENABLE Enable to disable memory ma	27-JUL-1984
10	M7	1944	SET PROTECTION ON PAGES.	27-JUL-1984
10	D8	1948	CALCULATE PTE ADDRESS SUBROUTINE	27-JUL-1984
10	F8	1950	DSX$MAPDBGBLOCK - MAP A BLOCK OF MEMORY	27-JUL-1984
10	G8	1951	DSX$FREEDBGSYM - FREE MEMORY MAPPED TO D	27-JUL-1984
10	H8	1952	MAP_DEBUGGER - MAP THE DEBUGGER'S MEMORY	27-JUL-1984
10	I8	1953	UNMAP_DEBUGGER - UNMAP THE DEBUGGER'S ME	27-JUL-1984
10	L8	1956	Symbol table	27-JUL-1984
10	D9	1961	Cross reference	27-JUL-1984
10	M9	1970	- TU81 LOAD DRIVER	27-JUL-1984
10	C10	1973	DECLARATIONS	27-JUL-1984
10	F10	1976	TU81 load device initialization	27-JUL-1984
10	I10	1979	TU81 load driver QIO	27-JUL-1984
10	M10	1983	Symbol table	27-JUL-1984
10	N10	1984	Psect synopsis	27-JUL-1984
10	B11	1985	Cross reference	27-JUL-1984
10	F11	1989	*** ODS Disk RMS routines	27-Jul-1984
10	G11	1990	declarations	27-Jul-1984
10	I11	1992	Advance RFA	27-Jul-1984
10	K11	1994	Access file block	27-Jul-1984
10	B12	1998	ODS OPEN service	27-Jul-1984
10	E12	2001	ODS$GET routine	27-Jul-1984

Fiche	Frame	Sequence		
10	J12	2006	ODS$READ routine	27-Ju -1984
10	N12	2010	*** ONLY730 NEBULA specific routines	27-JUL-1984
10	D13	2013	Macro invocations	27-JUL-1984
10	F13	2015	DSP$CONFIG_ADP Build P-table for adapter	27-JUL-1984
10	I13	2018	MAP$IOSPACE Validate mapping for adapter	27-JUL-1984
10	K13	2020	FETCH_LONG Retreive longword from possib	27-JUL-1984
10	L13	2021	PURGE_DATAPATH	27-JUL-1984
10	N13	2023	UBA$INITIAL UNIBUS ADAPTER INITIALIZATIO	27-JUL-1984
10	B14	2024	IO$TESTCSR_x Test CSR for existence	27-JUL-1984
10	C14	2025	IOGEN$CONN_VEC	27-JUL-1984
10	D14	2026	QIOCLEAN_CPU	27-JUL-1984
10	E14	2027	ONLY_SCB_CPU specific SCB setup	27-JUL-1984
10	F14	2028	SCBVECTOR_xxx Otherwise undefined SCB ve	27-JUL-1984
10	H14	2030	ONLY_CLOCK_INIT Perform cpu-specific TOD	27-JUL-1984
10	J14	2032	Symbol table	27-JUL-1984
10	L14	2034	Psect synopsis	27-JUL-1984
10	M14	2035	Cross reference	27-JUL-1984
10	F15	2041	ASCII STRING PARSE ROUTINE.	27-JUL-1984
10	I15	2044	DECLARATIONS	27-JUL-1984
10	J15	2045	ASCII STRING PARSE ROUTINE.	27-JUL-1984
10	E16	2053	SCAN$SYMBOL Scan symbol	27-JUL-1984
10	G16	2055	SCAN$COMPARE Compare strings	27-JUL-1984
10	I16	2057	SCAN$NUMERIC Scan number	27-JUL-1984
10	L16	2060	SCAN$SEARCHLIST Search list for unique s	27-JUL-1984
11	C1	2062	SCAN$DEVICE Scan a device name	27-JUL-1984
11	E1	2064	Breakup File name into parts	27-JUL-1984
11	G1	2066	Symbol table	27-JUL-1984
11	I1	2068	Cross reference	27-JUL-1984
11	B2	2074	PCSLOAD Module	27-Jul-1984
11	D2	2076	Libraries	27-Jul-1984
11	E2	2077	Table of Contents	27-Jul-1984
11	F2	2078	Psect Definitions	27-Jul-1984
11	G2	2079	PCSLOAD-Static Storage	27-Jul-1984
11	H2	2080	DSX$LOADPCS Routine	27-Jul-198.
11	J2	2082	DSV$INITPCS Routine	27-Jul-1984
11	M2	2085	*** PRINT Formatted print routines	27-JUL-1984
11	D3	2089	Libraries, Macros, and Equated Symbols	27-JUL-1984
11	E3	2090	Work Psect Declarations	27-JUL-1984
11	F3	2091	Data Psect Declarations	27-JUL-1984
11	G3	2092	DSV$SetPage Routine - Set terminal page	27-JUL-1984
11	I3	2094	DSV$ShowPage Routine - Show terminal pag	27-JUL-1984
11	K3	2096	DSX$Type_Out Routine - Do actual print	27-JUL-1984
11	C4	2101	DSX$Print Routine - Load type code byte	27-JUL-1984
11	E4	2103	DSX$PrintX Routines - Extended print rou	27-JUL-1984
11	I4	2107	DSX$CvtReg Routine - Mnemonic register c	27-JUL-1984
11	B5	2113	Symbol table	27-JUL-1984
11	E5	2116	Psect synopsis	27-JUL-1984
11	F5	2117	Cross reference	27-JUL-1984
11	N5	2125	*** PROBE Check address accessibility	27-Jul-1984
11	C6	2127	DSX$PROBE	27-Jul-1984
11	F6	2130	ACC$HANDLER	27-Jul-1984
11	I6	2133	- UDA50 BOOT DRIVER	27-JUL-1984
11	K6	2135	DECLARATIONS	27-JUL-1984
11	N6	2138	UDA50 Bootstrap device initialization	27-JUL-1984
11	D7	2141	UDA50 Bootstrap driver QIO	27-JUL-1984
11	F7	2143	Symbol table	27-JUL-1984
11	G7	2144	Psect synopsis	27-JUL-1984

Fiche	Frame	Sequence		
11	H7	2145	Cross reference	27-JUL-1984
11	L7	2149	*** QA Check Routines	27-Jul-1984
11	M7	2150	Table of Contents	27-Jul-1984
11	N7	2151	Libraries	27-Jul-1984
11	B8	2152	Included Macros and Literals	27-Jul-1984
11	C8	2153	External Routines	27-Jul-1984
11	D8	2154	External Own Storage	27-Jul-1984
11	E8	2155	Psect Definitions	27-Jul-1984
11	F8	2156	Local Macro Definitions	27-Jul-1984
11	H8	2158	Module-Global Static Storage	27-Jul-1984
11	I8	2159	ASCIC Bindings	27-Jul-1984
11	L8	2162	QA$Normal_Start	27-Jul-1984
11	F9	2169	QA$Multiple_Pass	27-Jul-1984
11	J9	2173	QA$Loop_On_Test	27-Jul-1984
11	C10	2179	QA$Loop_On_Subtest	27-Jul-1984
11	J10	2186	QA$Run_Backwards	27-Jul-1984
11	I11	2198	End of Module QACHECKS	27-Jul-1984
11	J11	2199	*** Setting and Showing QA Flags	27-Jul-1984
11	L11	2201	Table of Contents	27-Jul-1984
11	M11	2202	Psect Definitions	27-Jul-1984
11	N11	2203	Literals	27-Jul-1984
11	B12	2204	Static Storage	27-Jul-1984
11	C12	2205	Global Bindings	27-Jul-1984
11	D12	2206	VRSetQA	27-Jul-1984
11	H12	2210	VRShowQA	27-Jul-1984
11	L12	2214	End of Module QAFLAGS	27-Jul-1984
11	M12	2215	*** QA Routines	27-Jul-1984
11	B13	2217	Linkage Definitions	27-Jul-1984
11	C13	2218	Table of Contents	27-Jul-1984
11	D13	2219	Libraries	27-Jul-1984
11	E13	2220	Included Macros and Literals	27-Jul-1984
11	F13	2221	External Routines	27-Jul-1984
11	G13	2222	External Own Storage	27-Jul-1984
11	H13	2223	Psect Definitions	27-Jul-1984
11	I13	2224	Local Macro Definitions	27-Jul-1984
11	K13	2226	Module-Global Literals	27-Jul-1984
11	L13	2227	Field Definitions	27-Jul-1984
11	M13	2228	Module-Global Static Storage	27-Jul-1984
11	N13	2229	Supervisor-Global Static Storage	27-Jul-1984
11	B14	2230	DSX$BRANCH	27-Jul-1984
11	L14	2240	QA$Abort_QA	27-Jul-1984
11	N14	2242	QA$Add_Forced_Error	27-Jul-1984
11	B15	2243	New	27-Jul-1984
11	J15	2251	QA$Add_QA_Error	27-Jul-1984
11	K15	2252	QA$Control_C_Handler	27-Jul-1984
11	M15	2254	QA$Dump	27-Jul-1984
11	K16	2265	QA$Find_User_Call_Frame	27-Jul-1984
12	D1	2269	QA$Init	27-Jul-1984
12	F1	2271	QA$Main	27-Jul-1984
12	D2	2282	QA$Print_Forced_Errors	27-Jul-1984
12	E2	2283	Dispose	27-Jul-1984
12	K2	2289	QA$Print_Overall_Errors	27-Jul-1984
12	E3	2296	End of Module QAMAIN	27-Jul-1984
12	F3	2297		27-JUL-1984
12	G3	2298	QIO	27-JUL-1984
12	H3	2299	07-18	27-JUL-1984
12	I3	2300		27-JUL-1984

Fiche	Frame	Sequence		
12	J3	2301	Libraries and Equated Symbols	27-JUL-1984
12	L3	2303	External Symbols	27-JUL-1984
12	C4	2307	EXE$QIO QIO DISPATCHING ROUTINE	27-JUL-1984
12	E4	2309	QIO$INITIALIZE Initialize QIO database	27-JUL-1984
12	G4	2311	QIO$CLEANUP Remove old QIO database	27-JUL-1984
12	J4	2314	QIO$ADDUNIT Add a unit to the QIO databa	27-JUL-1984
12	D5	2321	INI_MBADP BUILD ADP AND INITIALIZE MBA	27-JUL-1984
12	F5	2323	INI_UBADP BUILD ADP AND INITIALIZE UBA	27-JUL-1984
12	G5	2324	INI_DRADP BUILD ADP AND INITIALIZE DRA	27-JUL-1984
12	I5	2326	MAXIMIZE ACCESS MODE	27-JUL-1984
12	J5	2327	BUG$CHECK Bugcheck notification routine	27-JUL-1984
12	K5	2328	QIO$LOAD_DB Insert unit into data base	27-JUL-1984
12	D6	2334	DB_ERROR ERROR LOADING DATABASE	27-JUL-1984
12	F6	2336	IOGEN$CNTRL_INI Initialize a controller	27-JUL-1984
12	G6	2337	MEMORY ALLOCATION SUBROUTINES	27-JUL-1984
12	H6	2338	ADPLINK Link adapter to end of ADP list	27-JUL-1984
12	I6	2339	Exe$PwrTimChk - Check reasonable interva	27-JUL-1984
12	K6	2341	Symbol table	27-JUL-1984
12	D7	2347	Psect synopsis	27-JUL-1984
12	E7	2348	Cross reference	27-JUL-1984
12	B8	2358	*** QIOFDT function dispatch routines	27-JUL-1984
12	E8	2361	ONE PARAMETER FUNCTION PROCESSING	27-JUL-1984
12	F8	2362	ZERO PARAMETER FUNCTION PROCESSING	27-JUL-1984
12	G8	2363	READ AND WRITE FUNCTION PROCESSING	27-JUL-1984
12	H8	2364	READ AND WRITE FUNCTION BUFFER CHECK AND	27-JUL-1984
12	I8	2365	READ AND WRITE BUFFER CHECK AND LOCK AND	27-JUL-1984
12	K8	2367	CHECK BUFFER ACCESSIBILITY FOR READ FUNC	27-JUL-1984
12	L8	2368	CHECK BUFFER ACCESSIBILITY FOR WRITE FUN	27-JUL-1984
12	M8	2369	CHECK BUFFER ACCESSIBILITY FOR READ FUNC	27-JUL-1984
12	N8	2370	CHECK BUFFER ACCESSIBILITY FOR WRITE FUN	27-JUL-1984
12	B9	2371	SET DEVICE MODE AND CHARACTERISTICS FUNC	27-JUL-1984
12	D9	2373	SENSE DEVICE MODE AND CHARACTERISTICS FU	27-JUL-1984
12	E9	2374	CARRIAGE CONTROL INTERPRETATION	27-JUL-1984
12	G9	2376	Symbol table	27-JUL-1984
12	J9	2379	Cross reference	27-JUL-1984
12	N9	2383	*** QIOREQ handle QIO request	27-JUL-1984
12	D10	2386	DECLARATIONS	27-JUL-1984
12	E10	2387	QUEUE I/O REQUEST	27-JUL-1984
12	J10	2392	COMPLETE I/O OPERATION	27-JUL-1984
12	K10	2393	QUEUE I/O PACKET TO DRIVER	27-JUL-1984
12	L10	2394	EXE$ALTQUEPKT - Call driver ALTSTART ent	27-JUL-1984
12	M10	2395	QUEUE I/O PACKET TO ACP	27-JUL-1984
12	N10	2396	INSERT I/O PACKET IN UNIT QUEUE	27-JUL-1984
12	B11	2397	INSERT I/O PACKET IN QUEUE BY PRIORITY	27-JUL-1984
12	C11	2398	Symbol table	27-JUL-1984
12	G11	2402	Psect synopsis	27-JUL-1984
12	H11	2403	Cross reference	27-JUL-1984
12	L11	2407	*** RELOCDRV relocate I/O driver	27-JUL-1984
12	N11	2409	INITIALIZE DRIVER DATA BASE	27-JUL-1984
12	B12	2410	REINITIALIZE DRIVER DATA BASE	27-JUL-1984
12	C12	2411	LOCAL SUBROUTINE TO EXECUTE OPERATION CO	27-JUL-1984
12	E12	2413	LOCAL SUBROUTINE TO LOCATE DATA STRUCTUR	27-JUL-1984
12	G12	2415	Symbol table	27-JUL-1984
12	I12	2417	Cross reference	27-JUL-1984
12	K12	2419	*** RMS Master RMS routines	27-Jul-1984
12	B13	2423	Device support table	27-Jul-1984
12	C13	2424	Allocate pseudo RPB	27-Jul-1984

Fiche	Frame	Sequence		
12	J13	2431	Deallocate RPB	27-Jul-1984
12	K13	2432	Check device support tables	27-Jul-1984
12	D14	2438	load error status	27-Jul-1984
12	G14	2441	load XAB	27-Jul-1984
12	I14	2443	allocate cache blocks	27-Jul-1984
12	K14	2445	initialize static table	27-Jul-1984
12	L14	2446	clean up code	27-Jul-1984
12	N14	2448	load error code in RAB	27-Jul-1984
12	B15	2449	load FAB error code	27-Jul-1984
12	C15	2450	verify validity of FAB	27-Jul-1984
12	D15	2451	verify cblock	27-Jul-1984
12	F15	2453	verify validity of RAB	27-Jul-1984
12	G15	2454	RMS$OPEN routine	27-Jul-1984
12	C16	2463	RMS$CONNECT routine	27-Jul-1984
12	F16	2466	RMS routine	27-Jul-1984
12	I16	2469	RMS$READ routine	27-Jul-1984
12	L16	2472	RMS$DISCONNECT routine	27-Jul-1984
13	C1	2474	RMS$CLOSE routine	27-Jul-1984
13	F1	2477	*** RT11 Console media RMS routines	27-Jul-1984
13	G1	2478	declarations	27-Jul-1984
13	H1	2479	advance record pointers	27-Jul-1984
13	I1	2480	Access file block via cache	27-Jul-1984
13	L1	2483	RT-11 OPEN service	27-Jul-1984
13	C2	2487	RT11$GET routine	27-Jul-1984
13	H2	2492	RT11$READ routine	27-Jul-1984
13	K2	2495	SYSTEM CONTROL BLOCK	27-JUL-1984
13	B3	2499	DECLARATIONS	27-JUL-1984
13	E3	2502	INITIAL SYSTEM CONTROL BLOCK IMAGE.	27-JUL-1984
13	G3	2504	DSX$INITSCB INITIALIZE SYSTEM CONTROL BL	27-JUL-1984
13	J3	2507	DSX$SETVEC SET VECTOR ROUTINE	27-JUL-1984
13	M3	2510	DSX$CLRVEC RESTORE VECTOR ROUTINE	27-JUL-1984
13	B4	2512	CHECK_VECTOR Verify vector range	27-JUL-1984
13	D4	2514	DSX$SETIPL SET INTERRUPT PRIORITY LEVEL	27-JUL-1984
13	F4	2516	SOFT$INT Software interrupt vectors	27-JUL-1984
13	G4	2517	Symbol table	27-JUL-1984
13	J4	2520	Psect synopsis	27-JUL-1984
13	K4	2521	Cross reference	27-JUL-1984
13	F5	2529	*** SCRIPT Read from command stream	27-JUL-1984
13	I5	2532	Libraries and External Symbols : [12]	27-JUL-1984
13	J5	2533	External Symbol Declarations	27-JUL-1984
13	K5	2534	Macros and Equated Symbols	27-JUL-1984
13	L5	2535	Data Psect Declarations	27-JUL-1984
13	M5	2536	Work Psect Declarations	27-JUL-1984
13	N5	2537	Absolute Psect Declarations	27-JUL-1984
13	B6	2538	SCRIPT$INIT - Initialize script	27-JUL-1984
13	D6	2540	SCRIPT$OPEN - Initialize a script file	27-JUL-1984
13	I6	2545	SCRIPT$FLUSH, STOP, CONT - Terminate all	27-JUL-1984
13	K6	2547	SCRIPT$FINISH - Terminate script input f	27-JUL-1984
13	M6	2549	DS_GETLINE - Get program data input line	27-JUL-1984
13	L7	2561	COPY_IT - Copy and translate	27-JUL-1984
13	N7	2563	ADVANCE - Advance to next record in buff	27-JUL-1984
13	C8	2565	Symbol table	27-JUL-1984
13	G8	2569	Cross reference	27-JUL-1984
13	C9	2578	*** SHOWMEM Handle Show Memory command	27-Jul-1984
13	G9	2582	Is a program loaded, really?	27-Jul-1984
13	J9	2585	DSV$ShowMemory routine	27-Jul-1984
13	K9	2586	ShowMemory Routine	27-Jul-1984

Fiche	Frame	Sequence		
14	K10	2599	VRSETMEM	27-Jul-1984
14	M10	2601	*** START Handle START command	27-JUL-1984
14	E11	2606	Declarations	27-JUL-1984
14	I11	2610	VRStart and VRReStart Routines	27-JUL-1984
14	L12	2626	VRAbort and DSX$Abort Routines	27-JUL-1984
14	C13	2630	VRSupport Routines	27-JUL-1984
14	E13	2632	VRShowSections Routine	27-JUL-1984
14	G13	2634	DSI$TIME_AST Routine	27-JUL-1984
14	H13	2635	Symbol table	27-JUL-1984
14	L13	2639	Psect synopsis	27-JUL-1984
14	M13	2640	Cross reference	27-JUL-1984
14	J14	2650	*** TSBTDRIVR driver for TS11/TS04, TU80	27-JUL-1984
14	L14	2652	DECLARATIONS	27-JUL-1984
14	D15	2657	TS11/TS04, TU80 Bootstrap driver	27-JUL-1984
14	K15	2664	Symbol table	27-JUL-1984
14	M15	2666	Cross reference	27-JUL-1984
14	E16	2671	- SYSTEM SERVICE UNWIND PROCEDURE CALL S	27-JUL-1984
14	H16	2674	UNWIND PROCEDURE CALL STACK	27-JUL-1984
14	K16	2677	UNWIND CONDITION HANDLER	27-JUL-1984
14	L16	2678	CALCULATE VALUE OF SP BEFORE CALL	27-JUL-1984
14	B1	2679	Symbol table	27-JUL-1984
14	C1	2680	Cross reference	27-JUL-1984
14	E1	2682		27-JUL-1984
14	F1	2683	VER730	27-JUL-1984
14	G1	2684		27-JUL-1984
14	H1	2685	Symbol table	27-JUL-1984
14	I1	2686	Cross reference	27-JUL-1984
14	J1	2687	*** VMSDUMMY Fake VMS driver entry point	27-JUL-1984
14	L1	2689	History	27-JUL-1984
14	M1	2690	Declarations	27-JUL-1984
14	N1	2691	Return-only Dummy Routines : [Oct	27-JUL-1984
14	B2	2692	MUTEX Control Routines	27-JUL-1984
14	D2	2694	Symbol table	27-JUL-1984
14	E2	2695	Cross reference	27-JUL-1984

B 1 Symbol table
C 1 Cross reference
D 1 Cross reference
E 1
F 1 VER730
G 1
H 1 Symbol table
I 1 Cross reference
J 1 *** VMSDUMMY Fake VMS driver e
K 1 *** VMSDUMMY Fake VMS driver e
L 1 History
M 1 Declarations
N 1 Return-only Dummy Routines ;
B 2 MUTEX Control Routines
C 2 MUTEX Control Routines
D 2 Symbol table
E 2 Cross reference
F 2 Cross reference
G 2 Directory
H 2 Directory
I 2 Directory
J 2 Directory
K 2 Directory
L 2 Directory
M 2 Directory
N 2 Directory
B 3 Directory
C 3 Directory
D 3 Directory
E 3 Directory
F 3 Directory
G 3 Directory
H 3 Directory
I 3 Directory
J 3 Directory